Lyes Benyoucef
Bernard Grabot
Editors

Springer Series in Advanced Manufacturing

Artificial Intelligence Techniques for Networked Manufacturing Enterprises Management

Springer

Springer Series in Advanced Manufacturing

Series Editor

Professor D.T. Pham
Manufacturing Engineering Centre
Cardiff University
Queen's Building
Newport Road
Cardiff CF24 3AA
UK

Lyes Benyoucef · Bernard Grabot
Editors

Artificial Intelligence Techniques for Networked Manufacturing Enterprises Management

Springer

Lyes Benyoucef, PhD
INRIA Nancy-Grand Est
COSTEAM/LGIPM
ISGMP Bat. A
Ile du Saulcy
57000 Metz
France
lyes.benyoucef@inria.fr

Bernard Grabot, PhD
LGP/ENIT
47, Avenue d'Azereix
65016 Tarbes Cedex
France
bernard@enit.fr

ISSN 1860-5168
ISBN 978-1-84996-118-9 e-ISBN 978-1-84996-119-6
DOI 10.1007/978-1-84996-119-6
Springer London Dordrecht Heidelberg New York

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

Library of Congress Control Number: 2010926592

Cover design: eStudioCalamar, Figueres/Berlin

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Series Editor's Foreword

The globalisation of manufacturing has led to the need for new management and control approaches for coordinating spatially and temporally distributed operations.

We welcome this latest addition to our Advanced Manufacturing Series. The new book, which addresses the application of AI techniques to the management of networked enterprises, is edited by two experts in the area of production and operations management, Professor Lyes Benyoucef and Professor Bernard Grabot.

We are particularly pleased as Lyes and Bernard were our collaborators in the recently completed EU-funded Network of Excellence project on Innovative Production Machines and Systems (I*PROMS). We thank them for completing the book both as a contribution to the field of intelligent manufacturing systems and as an important deliverable of our Network.

Duc Pham
Editor, Springer Series in Advanced Manufacturing
Cardiff, United Kingdom
January 2010

Preface

The global economy and the recent developments in information and communication technologies have significantly modified the business organization of enterprises and the way that they do business. New forms of organizations such as extended enterprises and networked enterprises (also called supply chain networks) appear and they are quickly adopted by most leading enterprises. It is well known that "competition in the future will not be between individual organizations but between competing supply chains" (Simchi-Levi *et al.*, 2003). Thus, business opportunities are captured by groups of enterprises in the same network. The main reason for this change is the global competition that forces enterprises to focus on their core competences (i.e., to do what you do the best and let others do the rest). According to a visionary report of Manufacturing Challenges 2020 conducted in the USA (National Research Council, 1998), this trend will continue and one of the six grand challenges of this report is the ability to *reconfigure networked enterprises rapidly in response to changing needs and opportunities*. Although the resulting networked enterprises are more competitive, the tasks for planning, managing and optimizing are much more difficult and complex.

While alliance-like enterprise networks with the underlying supply network represent tremendous business opportunities, they also make the involved enterprises face greater uncertainties and risks. Firstly, networks or supply chains have to be modified or dissolved once the business opportunities evolve or disappear. Secondly, changes or major perturbations at one enterprise may propagate through the whole network to other enterprises and hence influence their performance. The evolution from single enterprise with a high vertical range of manufacture towards enterprise networks offers new business opportunities especially for small and medium enterprises that are usually more flexible than larger companies. However, in order to be successful, performance and expected benefits have to be carefully evaluated and balanced in order to become a partner of the right network of enterprises for the right task. All these issues have to be taken into account in order to find an efficient, flexible and sustainable solution. A promising approach for that purpose is to combine analytical methods and knowledge-based approaches, in a distributed context.

Artificial intelligence (AI) techniques have been used in multiple segments of the networked enterprises. They have taken a prominent role to integrate people, information and products across dynamic networked enterprise boundaries including management of various manufacturing, logistics and retailing operations such as purchasing, manufacturing, warehousing and distribution of goods. Decisions involving customer profiling, new product development, retail marketing, and sales patterns are immensely refined using business intelligence tools. Further, as such decisions have an impact on the overall network enterprise processes, it is important that business intelligence tools should also be linked to networked enterprise management applications.

This book aims to align latest practice, innovation and case studies with academic frameworks and theories. It will include the latest research results and efforts at different levels including quick-response system, theoretical performance analysis, and performance and capability demonstration, hoping to cover the role of emerging AI technologies in modeling, evaluating and optimizing networked manufacturing enterprises activities at different levels.

Sixteen chapters were selected after a peer review process. They were revised in accordance with the suggestions and recommendations from the reviewers. They address prominent concepts and applications of AI technologies in managing networked manufacturing enterprises.

Chapter 1, by E. Oztemel, provides information on intelligent manufacturing systems. It also includes an analysis on historical progress of manufacturing systems as well as a brief review of traditional manufacturing systems. Fundamental technologies of AI are reviewed in order to establish the baseline for intelligent manufacturing systems. Moreover, basic characteristics of intelligent manufacturing and respective architectures are provided. Some examples of the applications of intelligent manufacturing systems are also highlighted in this chapter.

Chapter 2, by A.J. Soroka, describes a networked system developed for automating the gathering, processing and management of product fault knowledge. Such a system can remove the need for manual processing of fault reports and the associated problems. A reactive agent architecture based upon the concept of finite state automata (FSA) is implemented using the Java programming language. It also shows that the FSA-based agent architecture is suitable for application in this particular problem domain, due to the reactive nature of an FSA.

Chapter 3, by H. Ding *et al.*, presents a multi-agent-based simulation tool with descriptions of the overall architecture, modeling elements, operational policies, etc. The tool has been used in a commercial project with a leading high-tech manufacturer. The complex relationships between service levels, inventory cost, transportation cost, and forecasting accuracy are well studied. The project results show that networked enterprises can really get better insight from such a quantitative analysis and would be able to identify solid opportunities for cost saving and performance improvement.

Chapter 4, by Ouzrout *et al.*, focuses on unexpected swings in demand and on unexpected exceptions (problem of production, problem of transportation, etc.), which are important coordination and communication issues in supply chain management. The chapter analyses some of the existing approaches and work and describes an agent-based distributed architecture for the decision-making process. The agents in this architecture use a set of negotiation protocols (such as firm heuristic, recursive heuristic, collaborative planning and forecasting replenishment negotiation protocol) to collectively make decisions in a short time. The architecture is validated on an industrial case study.

Chapter 5, by Keshari *et al.*, presents a conceptual framework of a web-services-based e-collaborative system to establish a real-time information-sharing platform for enabling collaboration among similar types of manufacturing firms. The main idea of the chapter is to integrate the process planning and scheduling activities with the proposed system considering outsourcing as a viable technique of enhancing machine utilization and system's performance. An illustrative example is considered that demonstrate the working mechanism of the proposed framework.

Chapter 6, by Ounnar and Pujo, proposes a new approach for customer-supplier relationship control, in which the partnership is considered in the context of an association of potential suppliers within a network: an isoarchic control model for a supply chain network based on a holonic architecture. The decision-making mechanism is produced using an autonomous control entity. An implementation of the simulation of such a system is done via a distributed simulation environment high level architecture. A case study is presented.

Chapter 7, by A. Dolgui *et al.*, discusses the problems of lot-sizing and sequencing under uncertainties for a single machine or a flow-shop. The core of the chapter is a case study for multi-product lot-sizing and sequencing of flow-shop lines under uncertainties. Two main types of uncertainties are considered: breakdowns (random lead time) and rejects (random yield). The objective is to maximize the probability that customer demands will be satisfied at the given due date. A mathematical model of the problem and some heuristic and meta-heuristic approaches are discussed.

Chapter 8, by M. Souier *et al.*, presents a comparative study of a group of meta-heuristics, including taboo search, ant colony optimization, genetic algorithms, particle swarm optimization, electromagnetic meta-heuristic, and simulated annealing, against the modified dissimilarity maximization method for a job-shop routing problem picked from literature. Five criteria are selected for performance evaluation and comparison, namely: production rate, machine utilization rate, material handling utilization rate, work in process, and cycle time. The numerical results demonstrate that PSO and GA generate the best results practically in all cases.

Chapter 9, by D. Sánchez *et al.*, proposes the hybridization of evolutionary algorithms, well known for their multi-objective capabilities, with a supply chain simulation module in order to determine the inventory policy (order-point or order-level) of a single product supply chain, taking into account two conflicting objectives: maximizing customer service level and minimizing total inventory cost. Four algorithms (SPEA-II, SPEA-IIb, MOPSO and NSGA-II) are evaluated on five different supply chain configurations to determine which algorithm gives the best results and makes the best use of the simulator. The results indicate that SPEA-2 favors a rapid convergence and that modifying its crossover or its archive truncation rule (variant SPEA-IIb) may improve the results even further.

Chapter 10, by D. D'Addona and R. Teti, considers the development and implementation of a multi-agent tool management system (MATMAS) for automatic tool procurement. The design, functioning, and performance of diverse flexible tool management strategies (FTMS) integrated in the MATMS are illustrated. The MATMS operates in the frame of a negotiation-based multiple-supplier network where a turbine blade producer (customer) requires dressing of worn-out cubic boron nitride grinding wheels from the external tool manufacturer. The diverse FTMS paradigms, configured as domain-specific problem-solving functions operating within the MATMS intelligent agent and holding the responsibility for optimum tool inventory sizing and control, are tested by tool inventory management simulations and compared with real industrial cases.

Chapter 11, by N. Rezg and S. Dellagi, presents two studies investigating new intelligent integrated maintenance and production or service strategies, which deal with complex reliability problems. The first study describes a sequential constrained linear-quadratic stochastic production-planning problem in which a random demand must be satisfied and the single machine is subject to random failure. A minimal repair is performed at every failure, with preventive maintenance actions scheduled according to the manufacturing system history so as to reduce the failure frequency. The second study proposes a failure law that can vary over time, not only with respect to the preventive maintenance actions, but also as a function of changes in operating and/or environmental conditions. The goal is to determine intelligently the number of preventive maintenance actions, which must be performed in order to minimize the cost under a threshold availability constraint.

Chapter 12, by T. Yoo *et al.*, combines global random search using nested partitions (NP) with statistical selection using optimal computing budget allocation (OCBA) to design an innovative algorithm called NP+OCBA. As a non-trivial illustration, the developed NP+OCBA algorithm is applied to the multi-pass scheduling problem. A new multi-pass scheduling framework is presented that minimizes the number of rules to be evaluated by using an NP method, and minimizes the total number of simulation replications by using OCBA method. The efficiency and effectiveness of the proposed NP+OCBA are demonstrated by comparing its performance with that of three methods, respectively equal allocation, stand alone OCBA and COMPASS.

Chapter 13, by B. Desmet *et al.*, discusses some intelligent solution approaches used to optimize safety stocks in networked manufacturing systems. These approaches are based on normal approximation models for the involved critical safety stock parameters. The proposed approximation models are first tested on small example systems like distribution systems and assembly systems, and then in generic networked manufacturing systems. Moreover, they are benchmarked with results obtained from discrete-event simulation. As the various simulations show, the proposed approximations prove to be rather conservative and provide good upper bounds on the required system safety stocks.

Chapter 14, by S. Karnouskos *et al.*, considers the problem of how to deploy web services on shop-floor devices to connect them to enterprise systems. More specifically, it considers the case of a centrally managed population of devices that are located at different sites, dynamic discovery of devices and the services they offer, near real-time cross-site interaction, their interaction with business processes and distributed system management. The results show that the dynamic nature of the shop-floor can be utilized efficiently to plan further production orders and even implement last-minute changes on the production line using real-time data (real-time reconfiguration based on the application needs).

Chapter 15, by A.W. Colombo *et al.*, illustrates an overview of the engineering approaches, methods and tools that have been specified and developed within the European Research and Development project SOCRADES (www.socrades.eu). The results of the first set of successful applications in the area of elechtromechanical assembly systems, extending the concepts to geographically distributed service-oriented production sites are demonstrated.

Chapter 16, by J. Barata *et al.*, addresses the problem of shop-floor agility presenting as fundamental cornerstone for true agility and responsiveness of an enterprise wiling to participate in highly dynamic collaborative organizations and supply chains. The feasibility of the architecture proposed is demonstrated in a pilot implementation in a near-real shop-floor. Emerging web standards such as DPWS are used to guarantee cross-layer/abstraction interoperability ensuring that the shop-floor reacts positively to adjustments in the supply chain.

We hope you will enjoy the result of these efforts.

References

Simchi-Levi D, Kaminsky P, Simchi-Levi E (2003) Designing and managing the supply chain: concepts, strategies and case studies. McGraw-Hill

National Research Council (1998) Visionary manufacturing challenges for 2020. Committee on visionary manufacturing challenges, board on manufacturing and engineering design, commission on engineering and technical systems. National academy press

Lyes Benyoucef
Metz, France
January 2010

Bernard Grabot
Tarbes, France
January 2010

Acknowledgments

We wish to place on record our special thanks to Series Editor Prof. D.T. Pham and Senior Editorial Assistant Miss Claire Protherough for their valuable guidance and support during the entire process of editing the book. We offer our thanks to the Springer editorial team for their active role and support.

We would like to thank all reviewers for providing in-depth comments and constructive criticisms, and the authors for contributing their high-quality manuscripts. Without your help it would have been impossible to produce this book. Finally, we thank the I*Proms members, a network of excellence for innovative production machines and systems funded by the European Commission under its Framework 6 Programme (www.iproms.org).

Many thanks to Dr. Ritesh Kumar Singh and Mr. Akram Zouggari, INRIA researchers, for their excellent support in reviewing the book.

We hope you will enjoy the result of these efforts.

Lyes Benyoucef and Bernard Grabot

Contents

Chapter 1
Intelligent Manufacturing Systems

E. Oztemel

Abstract This chapter provides information on intelligent manufacturing systems including an analysis on change and historical progress of manufacturing systems as well as a brief review of traditional manufacturing systems. Fundamental technologies of artificial intelligence are also reviewed in order to establish the baseline for intelligent manufacturing systems. Following that, basic characteristics of intelligent manufacturing and respective architectures are provided. Some examples of the applications of intelligent manufacturing systems are highlighted.

Keywords Artificial intelligence, manufacturing system, fuzzy logic, genetic algorithm, holonic manufacturing system

1.1 Introduction

Automation is one of the major indicators of the change in manufacturing. Machines behaving themselves not only decrease the cost but also produce the products to be more compliant with the needs and specifications of the customers. Although technologies such as flexible manufacturing systems provide various advantages, automation itself is not enough to provide competitive advantage. In most of the modern manufacturing facilities, machines are capable of making decisions and exhibiting intelligent behaviour. That is to say that the manufacturing-related activities starting from the design to product shipment are being carried out through intelligent manufacturing technologies. As traditionally known, manufacturing systems are the integrated combination of various functions such as design, process planning, production planning, quality assurance, storing and shipment etc. In each of these functions several activities are carried out. Intelligent behaviour is exhibited by the machines in all of these functions. This diverted the attention of the researcher to "unmanned factories" and there have been quite interesting studies and implementations along this line. In this chapter, a general overview

E. Oztemel (✉)
Marmara University, Engineering Faculty, Dept. of Industrial Engineering, Turkey
e-mail: ercan.oztemel@bte.mam.gov.tr

of the change in manufacturing and intelligent manufacturing systems with some example applications is provided.

Manufacturing organizations have been facing a very challenging environment with dynamic and increasing complexity of activities. There is still a need for more flexible and dynamically changing systems to cope with the market requirements. Intelligent manufacturing systems are capable of providing this flexibility with increasing performance. They can facilitate highly complex manufacturing systems as well as various degrees of functionality of products. As they can handle design changes as quickly as possible, they can easily adapt themselves to the changes in the market and satisfy customer requirements, which are most of the time too versatile. They can also embed new methodologies and technological progress without any problem. Short manufacturing cycle times, shorter supplier times, adaptation to the changing situations in a short time frame, consistent knowledge flow, etc. provide advantages to global economic competition for which most of the manufacturing organizations need to sustain. Intelligent manufacturing systems are proven to be an effective tool for assuring these advantageous.

Since complexity and functionality of the products are increasing and companies need to sustain advantage in heavy competitive markets, it is not possible to make proper and effective decisions without the help and support of computer-based manufacturing systems. Considering interrelated activities in various manufacturing units, the importance of intelligent systems becomes more obvious than ever before. Since they can foresee problems before they occur and provide respective remedies, intelligent manufacturing systems can be extremely useful in supporting the expected level of competitiveness.

Another aspect of intelligent manufacturing is that, the traditional systems cannot be easily expanded in accordance with the technological achievements. They are now facing very challenging tasks such as the following:

- Automation is possible, but the limited decision-making capabilities of the machines make it nearly impossible to develop learning.
- Re-engineering is not easily achievable.
- Distributed management is hardly possible.
- Decentralized management makes it difficult to sustain overall integration.
- Reusability of systems (especially software components) is not possible.
- Synchronization of material and information flows is always problematic.

Historical progress in manufacturing systems shows several breakthroughs. Among them is the distributed manufacturing system. Due to technological developments in both robotics and information technologies, it is possible to design manufacturing systems in different locations but working in an integrated manner. Intelligent manufacturing systems contribute a great deal of effort to sustain this integrity and make it possible to perform manufacturing functionalities in distributed environments. Sustaining intelligent manufacturing systems can therefore assure both individual decision-making as well as cooperation with other related functions. Combining the advantages of intelligent and distributed systems increases the efficiency and effectiveness of manufacturing systems. Brun and Portioli (1999) highlighted the following benefits of this combination:

1. Complex problems can be divided into more simpler ones and sorted out easily.
2. Uncertain data and knowledge can be tolerated and handled to some extent.
3. Adaptation to the environmental changes can be achieved in shorter times with more effective responses.
4. Borders of the manufacturing systems can be expanded according to the new requirements.
5. Problems of distributed manufacturing systems can be eliminated or sorted out.

Setting up a tie among flexibility, modularity and decentralized control in manufacturing systems makes it possible to utilize limited resources as effectively as possible and prevents delays, which increases the customer satisfaction (Cantamessa, 1997). Especially, intelligent agent technology can be utilized to devise distributed and intelligent manufacturing systems. This is reported to prevent most of the problems of traditional manufacturing systems (Shen and Norrie, 1999a, 1999b). After a complete overview of traditional manufacturing systems (Sects.1.2, 1.3, 1.4, 1.5), intelligent manufacturing systems will be discussed in terms of their architecture (Sect.1.6) and respective components, as well as possible benefits to be encountered upon implementation (Sects. 1.7, 1.8). Section 1.9 concludes the chapter with some perspectives for future work.

1.2 Traditional Manufacturing Systems

It is important to review the traditional manufacturing system in order to comprehend the content of intelligent manufacturing systems and to understand the differences between common approaches.

A manufacturing system receives inputs (raw material, knowledge, energy, human resources, etc.) and transforms those into a set of outputs over several processes as shown in Figure 1.1. A set of activities within each process should be managed carefully in order to create a successful manufacturing environment.

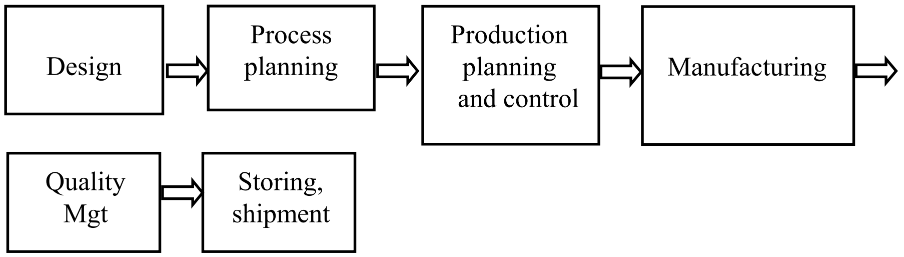


Fig. 1.1 Components of traditional manufacturing systems

Design refers to the process of creating a development plan for a product, structure, system, or component in such a way that specifications (shape, colour, dimensions, material composition, quality characteristics, part combination, etc.) are satisfied with minimum cost and time and that several criteria such as simplification, complying with standards, reliability, maintainability, safety, etc. are assured. A design process may include set of activities, some of which are listed below, depending on the product or service to be produced.

- Pre-production design:
  - defining and analysing the design goals;
  - specifying required solution and defining a set of requirements;
  - creating alternative design solutions;
  - selecting the best (or good enough) solution among the alternatives.
- Design during production:
  - implementing the selected design solution and perform improvements wherever possible;
  - testing the design solution.
- Post-production design:
  - introducing designed solution into manufacturing environment;
  - performing a pilot study and prototype production;
  - creating suggestions for improvements.
- Re-design:
  - improving the current design at any time before, during, or after production.

Process planning covers the selection of processes, equipment and tooling, and the sequencing of operations required by the design process. It is important to define the type of the manufacturing system and related processes. They have severe effects on all of manufacturing related decisions. A typical manufacturing system may be discrete where the system can handle intermediate steps or continuous where no interference is possible until the complete production cycle is implemented. The following manufacturing processes are possible.

- *Fixed production*: products are produced as per the requirement and specification of the customer and delivery schedule as well as the respective costs is fixed prior to the contract.
- *Batch production*: only a limited amount of each type of product is produced on the same set of machines.
- *Mass or flow production*: larger amounts of the same product are manufactured at a time. Since one line can produce only one type of product, this process is also called flow line production.

The main issues, which have an effect on the process plans, can be listed as the following:

1. effect of volume/variety of production;

2. capacity of the plant;
3. lead time;
4. required flexibility and efficiency.

Production planning and control includes decisions on production and inventory quantities. It generally consists of the planning of routing, scheduling, dispatching, inspection, and coordination and control of materials, methods, machines, tools and operating times. The ultimate objective is to define the organization of the supply and movement of materials and labour, and machine utilization and related activities in order to bring about the desired manufacturing results in terms of quality and quantity in expected time and place.

Production planning is usually done at an aggregate level, for both products and resources. Products to be manufactured are combined into aggregate product families that can be planned together so as to reduce planning complexity. Similarly production resources, such as machines or man-hours, are aggregated. While performing this aggregation, specific attention is required to assure that the resulting aggregate plan can be reasonably disaggregated into feasible production schedules.

Production planning is considered differently in various levels of the organizations. At the operational level production scheduling takes place. In order to do so, there is a "need to know" capacity and available resources as well as master production plans indicating the overall production amount for a certain planning period. The master plan should be generated from an aggregate planning which will in turn be based on demand forecasts for particular set of products (see Figure 1.2).

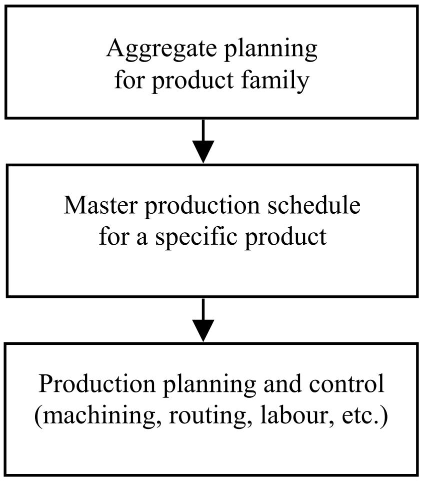

Aggregate planning for product family
Master production schedule for a specific product
Production planning and control (machining, routing, labour, etc.)

Fig. 1.2 Hierarchy of production planning activities

Manufacturing is transforming the inputs into respective products through several processes as designed in accordance with the process and production plans.

Quality management is a systematic process that translates quality policy into measurable objectives and requirements, and identifies the sequence of steps for realizing them within a specified time period. It includes creating and managing methods, criteria and techniques that ensure an output fits the intended purpose, and meets a stated expectation and is accepted by the programme. It assures that the products produced are within the tolerance limits and shifted to the customer as error free. It also includes making plans and corrections on the process in such a way that faulty products are not produced at all. Quality management mainly includes tree basic set of activities including:

- quality planning;
- quality assurance; and
- quality control.

Each of this set of activities intends to sustain the overall quality of the manufacturing processes as well as products produced by those processes.

Storing and shipment includes keeping the products in stock or transferring them to the customers. Detailed explanations of these, their sub-components and other components of manufacturing systems can be found in Chryssolouris (2006).

1.3 Changes in Manufacturing Systems: a Historical Perspective

Changes in manufacturing systems indicate the historical progress still continues. It is important for the companies and manufacturers to follow the changes in order to keep up with the technological progress and be capable of satisfying the expectations of the customers, which are mainly based upon the contemporary understandings. Due to several factors such as progress in technology, the nature of manufacturing systems has been changing from one form into another such as from manual systems to fully automated and autonomous systems. It is important to note that the changes are occurring more frequently than ever before. It is therefore not easy to follow up the change as it was in the past. It requires not only setting up the new machines but also to know every detail and possible projections towards the future.

It is also important to realize that the progress in manufacturing systems is not only related to the machines and technology but also to several other aspects including:

- manufacturing methods;
- environmental changes;
- managerial changes; and
- customer expectations and requirements.

Technological progress includes the changes in manufacturing technologies and respective infrastructure over the time horizon. Figure 1.3 shows the historical perspective along this line.

$$\begin{tikzcd}
\text{Manual Systems} \arrow[dr] & & & & & \\
& \text{Semi-manual Systems} \arrow[dr] & & & & \\
& & \text{Traditional Machines} \arrow[dr] & & & \\
& & & \text{Automated Machines} \arrow[dr] & & \\
& & & & \text{Intelligent Machines} \arrow[dr] & \\
& & & & & \text{Autonomous Machines (holons)}
\end{tikzcd}$$

Fig. 1.3 Changes in machining systems

As indicated in Figure 1.3, machining operations are transformed themselves into autonomous systems, which are capable of handling manufacturing related activities taking all aspects of manufacturing, and environmental issues into account. The respective changes are in progress and it is not difficult to foresee fully unmanned factories and virtual machines dominating manufacturing sites.

In parallel to the progress on machining systems, new products and production processes are also being established in the following areas:

- automation and information systems;
- rapid prototyping;
- intelligent and autonomous manufacturing systems;
- intelligent materials;
- nano and biotechnologies;
- innovative and R&D processes and products.

In order to sustain competitive advantage, the progress in these areas should be carefully monitored and manufacturing systems should be redesigned to cope with those.

Changes in manufacturing methods are also apparent in manufacturing systems. Figure 1.4 indicates possible progress along this line.

As seen in Figure 1.4, manufacturing methods are emerging towards fully automated and unmanned manufacturing systems which could be highly flexible, reconfigurable, reusable, and interoperable as well as autonomous and intelligent. Tendency to evolve in this aspect still continues to increase with holonic manufacturing systems.

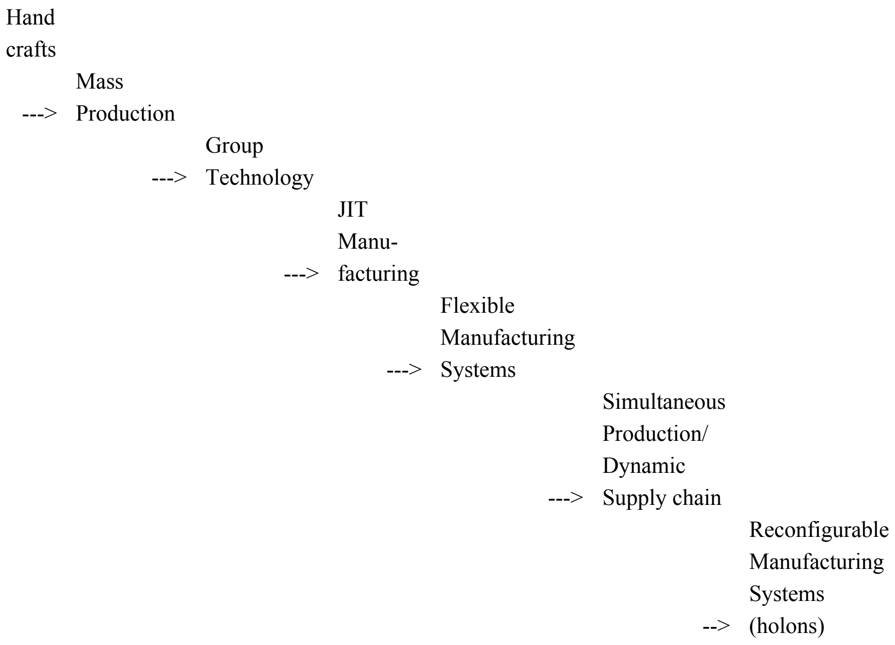

Hand crafts
---> Mass Production
---> Group Technology
---> JIT Manu-facturing
---> Flexible Manufacturing Systems
---> Simultaneous Production/ Dynamic Supply chain
--> Reconfigurable Manufacturing Systems (holons)

Fig. 1.4 Changes in manufacturing methods

The progress in methodologies of manufacturing systems also triggers new approaches in manufacturing planning systems and their management. Figure 1.5 illustrates the evolution of the respective systems.

Among these, especially lean manufacturing, where limited resources are used as much effectively and efficiently as possible (see Bamber and Dale, 2000) and agile manufacturing, which can be considered as the capability of manufacturing systems to adapt themselves to unexpected or unanticipated situations (see Sanchez and Nagi, 2001) became very popular and attracted significant attention in academic and business environments.

Environmental changes include ecological developments in manufacturing areas. Responsible bodies make every effort to sustain the environmental safety and provide regulations, which binds manufacturing activities as well. Global warming, air pollution, environmental factor analysis, etc. are the main concern of manufacturing facilities. Intelligent systems may be very useful to handle the requirements and to keep the manufacturing locations complying with environmental regulations. This could surely be achieved through implementing environmentally safe manufacturing methodologies and technologies.

Managerial changes are another concern, which has a potential effect on manufacturing organizations. Figure 1.6 illustrates the progress in managerial approaches since the beginning of the industrial revolution. As indicated the managing organizations faced dramatic changes over the last 100 years.

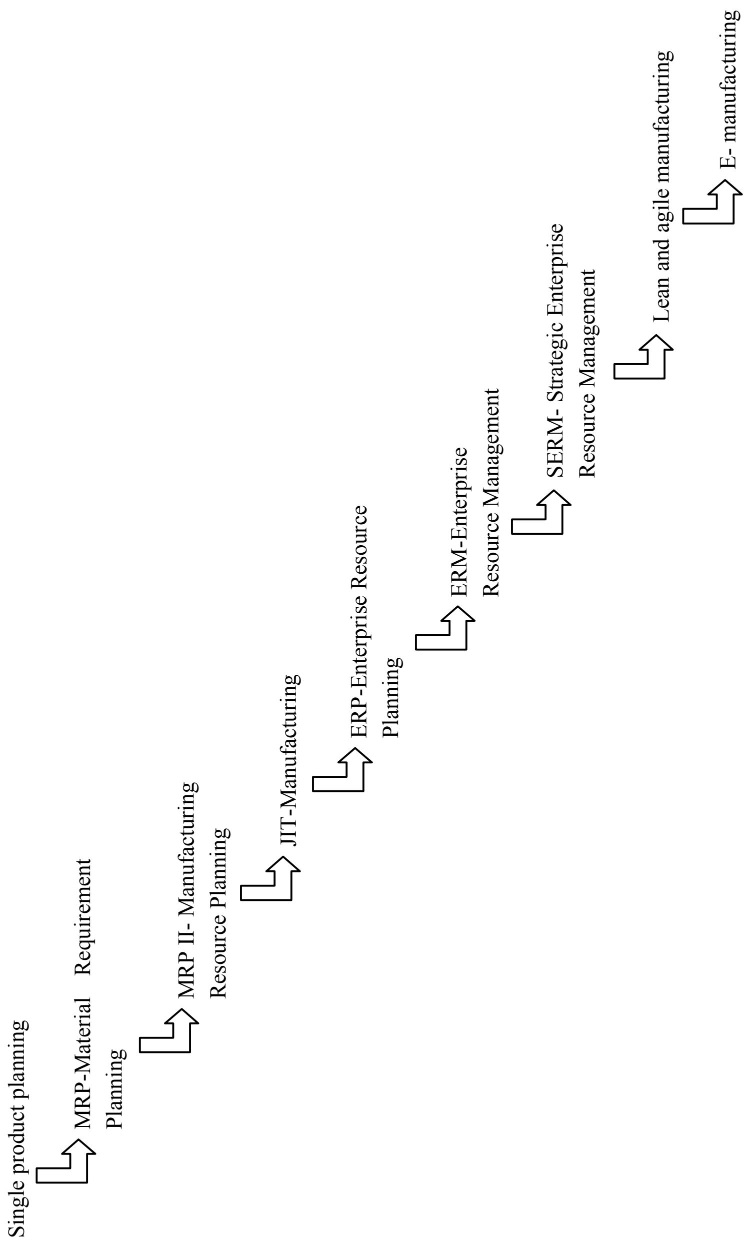

Single product planning
MRP-Material Requirement Planning
MRP II- Manufacturing Resource Planning
JIT-Manufacturing
ERP-Enterprise Resource Planning
ERM-Enterprise Resource Management
SERM- Strategic Enterprise Resource Management
Lean and agile manufacturing
E- manufacturing

Fig. 1.5 Changes in manufacturing planning and management systems

Workshop management

--> Factory shop management

--> Functional management

--> Process management

--> Performance based management

--> Management by objectives

--> Strategic management

Fig. 1.6 Changes in management approaches

Changes are not only bounded by technology, management or other aspects of manufacturing but also by *customer requirements and expectations*. New manufacturing methods and technologies led to new products with more functionality and more capability to serve the needs of the customer. Some products even diverted the attention of the customers' new expectations, which would in turn drive the new approaches to be implemented in manufacturing systems. This triggered the competition and became one of the basic elements of competitive advantage. Manufacturing organizations cannot only be proud of satisfying their customers anymore. They also need to foresee the future needs of their customers and try to be ready to serve them when the need in question is experienced. It is also recommended to the organization not to wait but to create new expectations to sustain competitive advantage. In the old days, customers used to buy whatever they found in the market. This is not the case anymore. They want to buy products that are capable of satisfying them both today and in the future. Figure 1.7 illustrates the changes in customer expectations.

Customer buys

--> Whatever available in the market

--> Whatever he/she wishes to buy

--> Through selecting among alternative products

--> Products with multifunctional capabilities

--> Products satisfying his future needs

Fig. 1.7 Changes in customer requirements and expectations

Similarly to others, intelligent systems can be considered as one of the useful ways of dealing with customer expectations. Asking customers to design the products they want to purchase and producing exactly the same designs are not fantasy anymore. Integrating virtual reality and manufacturing systems can easily make this happen. This is already being experienced in various companies all over the world.

As explained above, the world is facing a very challenging change and requires manufacturing organizations to cope with those in order to be successful and gain competitive advantage. This necessitates overall integration of manufacturing functions from design up to product shipment. This integration is facilitated through information technology networks with respective information management systems. Figure 1.8 shows the basic components of the integrated manufacturing systems.

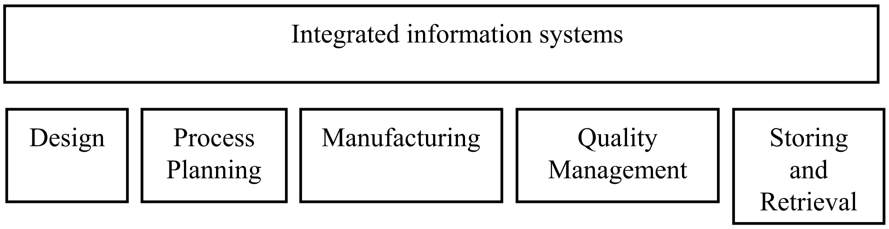


Fig. 1.8 Basic functions of an integrated manufacturing organization (Kusiak, 2000)

Developments in computing systems did not only make information systems capable of integrating manufacturing functions but also created important progress in performing the respective functions as well. Design activities are facilitated through computer-aided design (CAD) systems. Similarly, computer-integrated manufacturing (CIM), computer-aided manufacturing (CAM), computer-aided process planning (CAPP), computer-aided quality control (CAQC) and automated storage and retrieval (AS/R) systems and automated guided vehicles (AGV) are developed with the help of computing systems. Effective utilization of computers in manufacturing created numerous advantages including the increase in productivity and quality as well as flexibility and decrease in lead times, work in process inventories, and costs (Hannam, 1997). Figure 1.9 highlights the progress from very basic traditional manufacturing up to intelligent manufacturing systems.

As can clearly be seen, there are various levels of manufacturing technologies implemented for manufacturing processes. Figure 1.10 illustrates different aspects of manufacturing system with respect to various technologies implemented.

The analysis provided so far brings two important issues into attention. They are the automation and intelligent systems. Both are essential components that drive the changes experienced. They are in fact not only triggering the change but also increasing the speed of the change as well. Changes experienced in every 15–20 years before now can be achieved within 3–5 years. Automation and intelligent systems together develop their performance in manufacturing at a rapidly increasing speed. The following sections of this chapter are devoted to explaining intelligent manufacturing systems and their applications. For the sake of completeness a brief introduction to artificial intelligence (AI) is provided first.

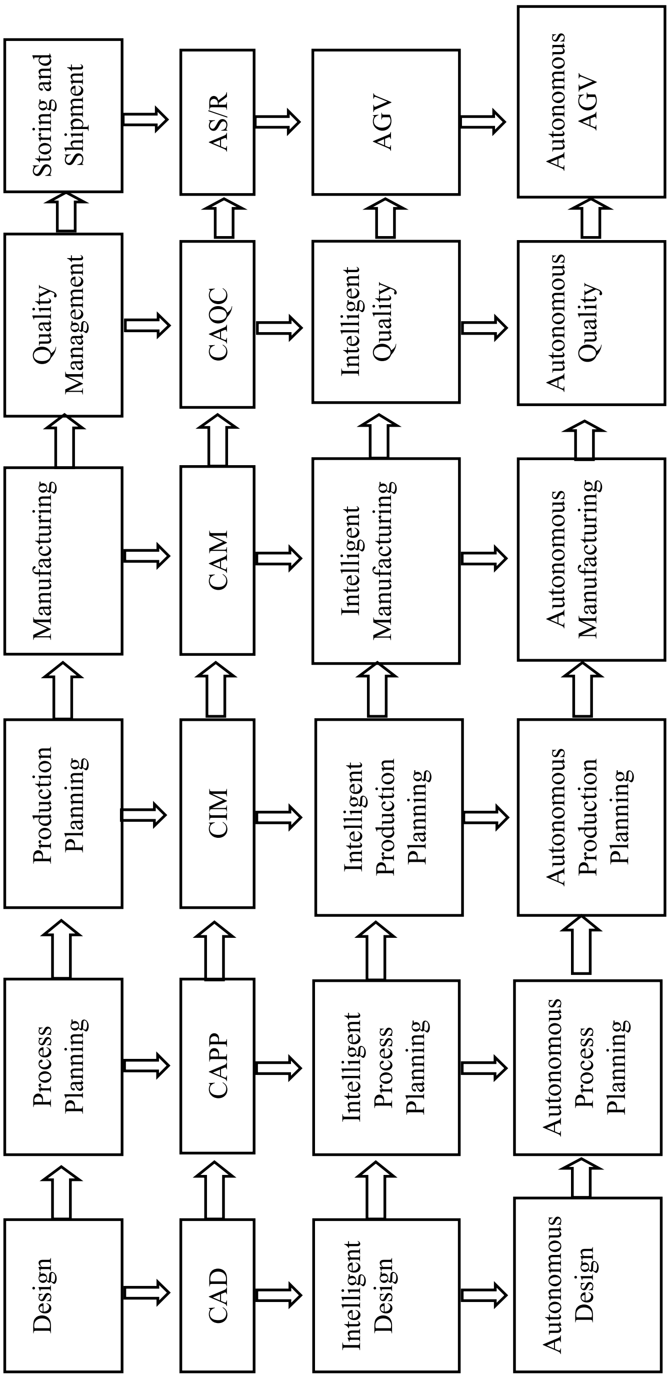


Fig. 1.9 Changes and progresses in manufacturing systems

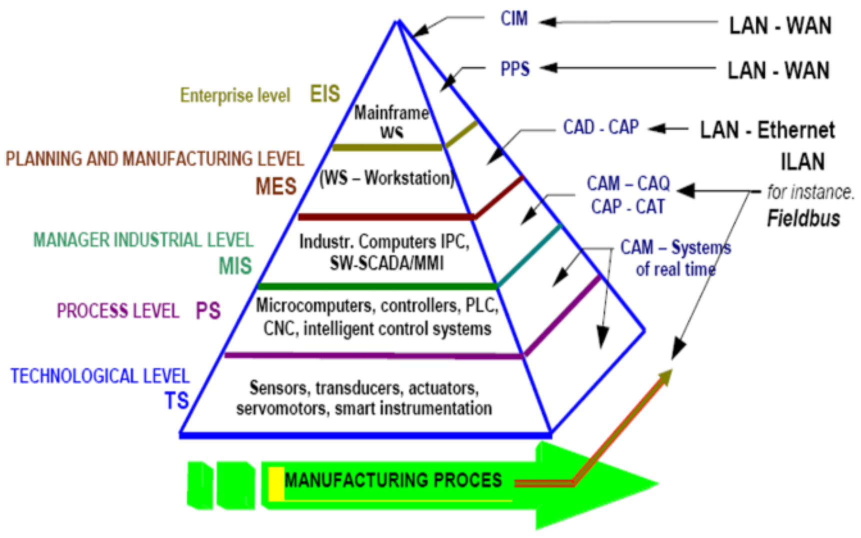

CIM
LAN - WAN
PPS
LAN - WAN
Enterprise level EIS
Mainframe WS
CAD - CAP
LAN - Ethernet
ILAN
PLANNING AND MANUFACTURING LEVEL MES
(WS – Workstation)
CAM – CAQ
CAP - CAT
for instance.
Fieldbus
MANAGER INDUSTRIAL LEVEL MIS
Industr. Computers IPC, SW-SCADA/MMI
CAM – Systems of real time
PROCESS LEVEL PS
Microcomputers, controllers, PLC, CNC, intelligent control systems
TECHNOLOGICAL LEVEL TS
Sensors, transducers, actuators, servomotors, smart instrumentation
MANUFACTURING PROCES

Fig. 1.10 Hierarchical multilevel structure of manufacturing enterprise (Smutny, 2003)

1.4 Artificial Intelligence and Intelligent Manufacturing Systems

Intelligent manufacturing systems are those performing the manufacturing functions as if the human operators are doing the job (Kusiak, 2000). They are equipped with a sufficient level of intelligence to perform such activities. However, it is important to note that none of the intelligent systems is capable of replacing their human counterparts. They are designed to support the operators, not to replace them. They can provide knowledge and respective facts filtered out from the huge amount of stored knowledge and increase the speed of decision-making capability of the operators. They can utilize decision-making capability of computers and make it available to the operators. Since it is necessary to equip the systems with intelligent capabilities in order to accomplish this, it would be wise to briefly review AI technologies at this point.

1.4.1 Technologies of Artificial Intelligence

Although there is no standard definition for AI, it is generally accepted as the branch of computer science dealing with intelligent behaviour and trying to make computers capable of exhibiting it. This necessitates different computing technologies as opposed to traditional approaches. For example, knowledge is processed instead of data. Traditional algorithms are replaced by heuristic algorithms; and instead of numeric representation, symbolic representation is taken as the main focus. Moreover, AI systems may make mistakes as humans would do and can exhibit as much intelligence as their developers. Since they make decisions on the basis of the available knowledge, if the knowledge is not right, the respective decisions would not naturally be erroneous as well. The decisions may obviously be considered to be correct by those who provide the knowledge but wrong by those who disagree with that particular knowledge. Disagreement between experts or knowledge owners is always known to appear in every area. Another aspect, which makes AI different from the other computing systems, is that they can perform actions with uncertain or even inexact and incomplete knowledge.

In order to better understand AI, the concept of "intelligence" should be carefully analysed and understood. Intelligence is to perceive the environment in which the system is operating, to relate events taking place around the system, to make decisions about those events, to perform problem solving and generate the respective actions and control them (Meystel and Albus, 2001). In order to achieve these, the computers or systems should be equipped with the capability of planning, learning, reasoning, monitoring, control, etc. Several methodologies are developed to create these capabilities. Several of them are highly effectively used in manufacturing systems. Note that there are various technologies, which are still in laboratories, and heavy academic research is carried out on them. Some AI techniques, however, show remarkable progress and are utilized in every area affecting human life. Among those that are popularly known are:

1. expert systems;
2. artificial neural networks;
3. genetic algorithms;
4. fuzzy logic; and
5. intelligent agents.

Brief description of each of these is provided below. The readers are recommended to review the respective references for detailed information on each of these.

1.4.1.1 Expert Systems

Expert systems are widely used in most of the industrial applications. Manufacturers may definitely exploit advantages of this technology for their own businesses. Expert systems make it possible to create intelligent software systems capable of solving problems in the same way as human experts would do when facing the same problems. Similarly, to human experts, they utilize their knowledge, experience and talents. This is possible when respective knowledge is stored in a specific format understandable to the computer. The knowledge, which is represented in machine-readable format, is stored in so-called *knowledge bases*. It is utilized by expert systems for a decision-making process similar to that of a human being in order to produce solutions to the problems. Figure 1.11 indicates basic components and architecture of an expert system. Each component is briefly explained below.

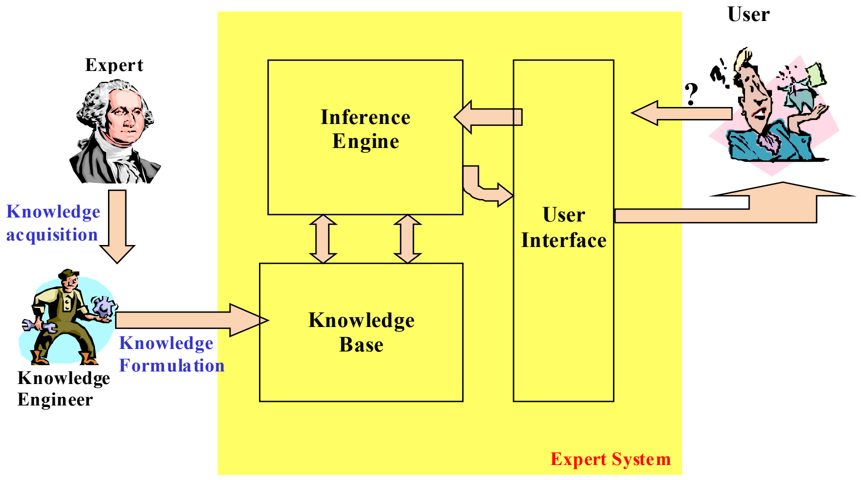


Fig. 1.11 Components of an expert system

- *Knowledge acquisition*: an expert system cannot be called an expert system unless it possesses the knowledge of at least one expert of a specific domain. It is necessary to acquire and elicit respective knowledge from the expert and represent that to the computer in a machine-readable format. This process is called knowledge acquisition.
- *Knowledge base* is the place where acquired knowledge is stored. Traditional ways of knowledge representation may include using IF…THEN… rules, knowledge frames, classes and procedures. The knowledge base is populated with the domain knowledge using these formats. Some other knowledge representation approaches can be used depending on the nature of the knowledge available.
- *Inference engines* search and scan the knowledge base, filter the required knowledge out and reason about those in order to be able to solve the problem posed by the user through the user interface. There are two general approaches for inferencing.
  - *Forward chaining*: this starts with the facts of the domain and tries to achieve a solution through available knowledge in the knowledge base.
  - *Backward chaining*: this starts with a solution and tries to find out respective facts supporting that particular solution.

- *User interface* handles the communication between the expert system and the users. It may explain how and why certain conclusions are reached.

As indicated in Figure 1.11, the expert is one of the key elements of an expert system. There is a need for at least one expert to develop the expert system in his/her area of expertise. Knowledge engineers elicit knowledge from the expert and formulate it in such a way that the system can understand. Once the user is seeking for a solution to his/her problem, the inference engine scans the knowledge base and identifies the solution on the basis of the knowledge articulated by the expert and makes it available in the knowledge base. The content of the knowledge base may increase in time and the system may be able to solve more complex problems. However, it is important to identify and populate the knowledge base with very basic knowledge at the first place in order to start solving the problems. It is also important to note that the expert systems are capable of solving the problems, which are vague even to the manufacturing knowledge. They are proven to be useful especially in:

- planning;
- control;
- maintenance;
- repair;
- diagnosis;
- monitoring; and
- inspection, etc.

Detail descriptions of expert systems can be found in Turban *et al.* (2008).

1.4.1.2 Artificial Neural Networks

Artificial neural networks are mainly designed to perform learning. They can generate new knowledge through learning the manufacturing events using the examples. They are formed by hierarchically connected process elements capable of information processing. Each connection between the processing elements has a weight value indicating the effect of one processing element on the others. These weight values, once the network is trained, are believed to indicate the knowledge of the network. The main purpose of the learning is therefore to find out the right weight values of the connections. Since the weight values are distributed to the overall network, the neural networks are assumed to have a distributed memory. Figure 1.12 shows an example of an artificial neural network.

Artificial neural networks, which created several new approaches to computing science including non-algorithmic, adaptive and parallel computing, proved that the computers could learn. They have various characteristics some of which may include the following:

- They can only work with numeric information.

- They can learn using examples.
- They have the capability to process incomplete information.
- They can perform graceful degradation and they are fault tolerant.
- They have distributed memory.
- They automatically generate information for unseen examples.
- They are mainly used for perceptual information processing.
- They perform learning well on pattern recognition and classification.
- They have a self-organizing capability.

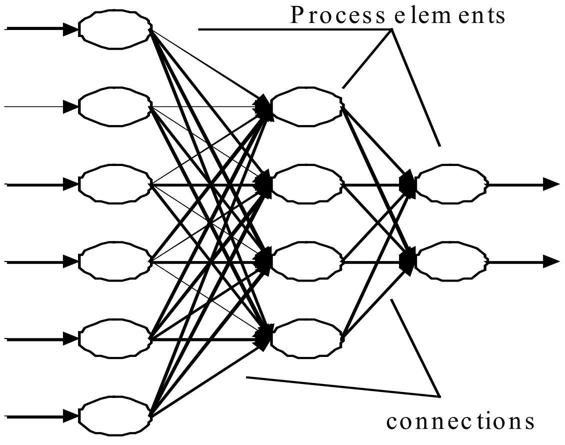


Fig. 1.12 An example of a neural network

Successful applications of artificial neural networks clearly indicate that they can be very effective tools for solving problems which are very noisy, multi-dimensional, non-linear, complex, uncertain, incomplete, or prone to errors provided that sensory information (examples) is available. Several examples of successful manufacturing applications can be realized especially in the following areas:

- probabilistic function estimation;
- pattern recognition and classification;
- pattern matching;
- time series analysis;
- signal filtering;
- data fusion;
- non-linear signal processing;
- non-linear system modelling;
- optimization;
- intelligent and non-linear control;
- data mining;

- optical character recognition;
- optimum path planning for robotics;
- fingerprint recognition;
- robot motions;
- life cycle estimation of mechanical parts;
- sonar signal classification;
- production planning and scheduling.

Detail information on neural networks and how to use them can be found in Bosque (2002). Moreover, applications in various areas are also edited by Rabunal and Dorado (2006).

1.4.1.3 Genetic Algorithms

Genetic algorithms are very popular in solving especially very complex optimization problems. A set of some initial solutions to a specific manufacturing problem is randomly selected. These initial solutions are paired as parents and new solutions are produced (children) out of them. In each production like this, the solutions are aimed to be improved. Once new solutions are generated, some of them are carried out to the next generation in order to produce new solutions. This process continues until no better solution is achieved. The better solutions are sought through genetic operators and a fitness function. Basic elements of a genetic algorithm, which are given in Figure 1.13, are explained below.

Chromosomes and genes indicate possible solutions. There could be *N* solutions for a problem (*N* chromosomes) in the solution space from which the best solution is sought. Each chromosome is formed by *genes*, which are the basic characteristics of the intended solution.

Initial population is the set of randomly selected chromosomes (solution pool). Depending on the type of the problem, there could be certain number of initial solutions required.

Crossover is a genetic operator making it possible to create new solutions using two existing ones. Crossover rate defines how many solutions to be parented.

Similarly, *mutation* is another operator, which is used to direct the searching process to different paths. This prevents going back to the solutions which were encountered before. Mutation rate defines how many chromosomes to be mutated.

Fitness function indicates how good the solution is for the given problem. There is a need to identify this function for every problem. This could be, for example, the total completion time in job-shop scheduling. In this case, each time new solutions are generated, completion time is calculated and new solutions are generated until the total completion time can no more be reduced.

Reproduction is to keep the solution space size fixed for computational efficiency. Whenever new solutions are generated the solution space is updated and some of the newly generated solutions go into the solution pool and some old ones are removed. Generally, the famous Russian-roulette methodology is implemented to reproduce the solution pool. However, there is no limitation in this respect and the designer may develop his own strategy and method for reproduction. Detailed descriptions of genetic algorithms can be found in Revees and Rowe (2004).

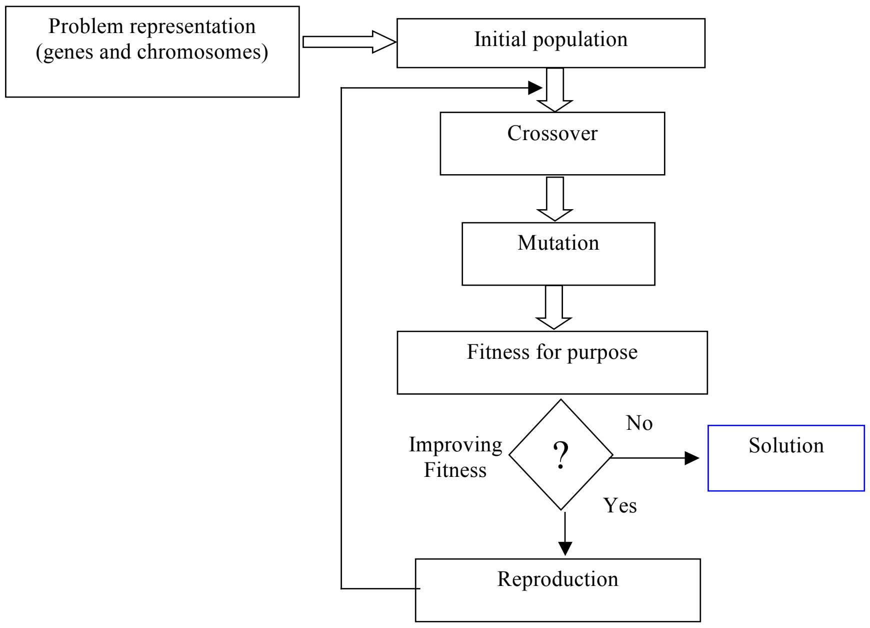


Fig. 1.13 Basic elements and procedure of genetic algorithms

1.4.1.4 Fuzzy Logic

Fuzzy logic is particularly developed to handle uncertain knowledge. Most of the decisions experts make have some sort of uncertainty in them. Experts tend to use vague phrases. Instead of saying for example, "*turn the thermostat off when the temperature of the machine is 25 degrees*", they prefer to direct using the phrases like "*turn the system off when the temperature is high***"**. The term high, which is called a *fuzzy variable*, may be regarded differently by different people. Some may consider 25 degrees high whereas others do not. Since fuzzy logic is dealing with linguistic expressions like this, it is also referred as *"computing with words"*. Another good thing with fuzzy logic is that it may take a continuum approach and produce solutions for all alternatives and values. For example, suppose that the temperature is to be turned off when the temperature is 25 degrees. What happens if the temperature is 24.99 degrees? Using the crisp logic the system will never turn it off. However, fuzzy logic starts turning the thermostat nearly off in this case as it takes temperature values from –∞ to +∞ into account. This is decided based on a value called *membership value*. This value indicates the possibility and degree of a particular instant to be the member of the fuzzy set. This value is between 1 and 0 where 1 indicates total membership and 0 indicates no membership at all. The designer for each particular fuzzy variable defines this function. Note that membership value does not indicate the *probability* but rather it is the *possibility*. Figure 1.14 provides basic components of fuzzy logic.

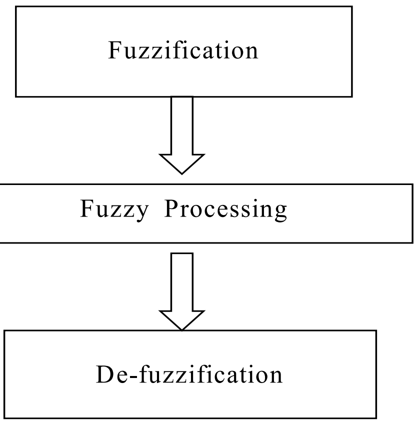


Fig. 1.14 Components of fuzzy logic

An example of a membership function for fuzzy variable "normal" is given in Figure 1.15. As can be seen, every temperature value between [–∞, ∞] has a membership value.

Fuzzification is to generate fuzzy propositions and identify linguistic variables and their membership functions. *Fuzzy processing* is to generate fuzzy rules and implement them in order to generate a solution space for the given problem. Generally membership functions are laid on top of each other and solution space is defined according to *AND* and *OR* operators of the rules as in traditional logic. Figure 1.16 shows an example of a solution space.

Defuzzification is the process of finding a crisp value for the solution. Fuzzy logic produces a solution space but this may not be applicable in most cases and a crisp value may be needed. There are various methods for defuzzification. Generally the value corresponding to the highest membership value is taken as the solution to the problem. This may be difficult for some solution spaces as there may be more than one value having the same membership value. In this case, the average of those, or a value corresponding to the geometric average of the solution space is taken as the net solution to the problem. Detailed information on fuzzy logic and its applications can be found in Cox (1999).

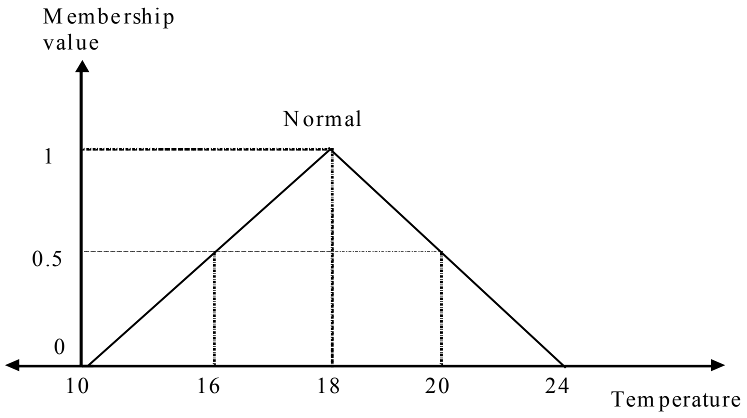


Fig. 1.15 An example of a membership function

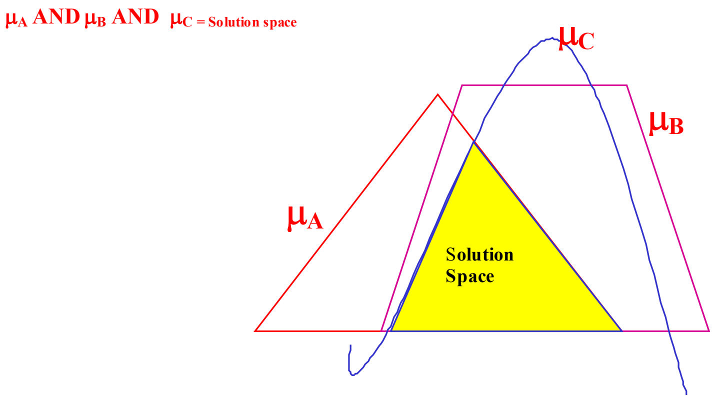


Fig. 1.16 An example of a solution space

1.4.1.5 Intelligent Agents

Intelligent agents are independent and autonomous systems in performing their intended functions. They can be both hardware and software systems and they may incorporate more than one AI technology. They can learn and work concurrently. The general architecture of an agent is given in Figure 1.17. As seen in Figure 1.17, there are three main components such as perception, cognition and action.

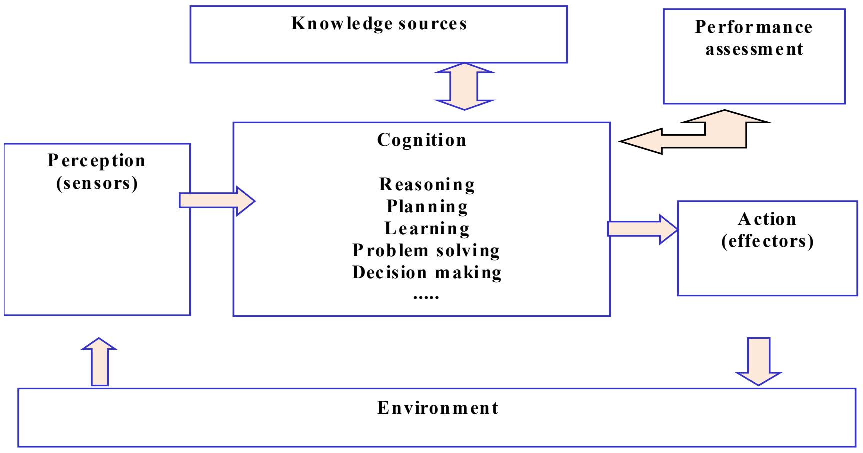


Fig. 1.17 General architecture of an intelligent agent

Perception is to receive the inputs from the environment through sensors and convey them to the cognition module to be processed. This process may include filtering, and prioritization according to the order of importance. *Cognition* is to process the information perceived and making decision accordingly. This process may require various methods of intelligence systems such as learning to be implemented. The cognition mechanism of an agent may also deal with unexpected situations and adapt itself to new situations as quickly as possible. Therefore a highly dynamic and flexible architecture needs to be established. *Action* is to perform the command received from the cognition using the respective effectors. This could for example be to perform walking for a robot or stopping when an immediate barrier in front is encountered. Once the agent acts, this may change the environmental signals and new information is generated. This can immediately be perceived by perception and respective action is generated for the effectors to operate.

There have been various types of agents in the literature and details are provided by Russel and Norvig (2002). Similarly, Wooldridge (2009) has provided information on multi-agent systems.

1.4.2 Intelligent Manufacturing Systems

Intelligent manufacturing systems are those utilizing AI techniques for manufacturing activities. They can exhibit all characteristics of intelligent systems such as learning, reasoning, decision-making, etc. They can utilize several AI technologies in order to perform their intended functions. Intelligent manufacturing systems can be designed in such a way that they can operate when it is difficult to measure the outcomes, where frequent changes in operations are possible and when there are no available prior decisions about the system behaviour (Rzevski, 1997).

Basic characteristics of intelligent manufacturing include the following:

- They minimize human involvement in manufacturing activities.
- They arrange material and production compositions automatically wherever possible.
- They monitor and control production processes and manufacturing operations.
- They recommend and take immediate actions to prevent faulty production.
- They perform maintenance activities.
- They make sure that the processes are working properly and monitor their performance.
- They diagnose machines and sustain manufacturing integrity.

There are numerous examples of intelligent manufacturing systems developed and in use today. Although various approaches have been developed, the fundamental baseline remains the same. Kusiak (1990) provides a general overview of these systems and highlights basic properties.

Applications indicate that intelligent systems and manufacturing systems can be integrated in various ways. Sometimes existing systems are improved with the help of AI technologies. Sometimes intelligent and totally new systems are designed. In order to achieve this integration it is important to note that there is also a need for modelling and algorithm management systems as they are the essential baseline in developing components of the manufacturing systems. Generally three approaches as explained below can be realized in creating intelligent manufacturing systems.

1.4.2.1 Artificial-intelligence-supported Manufacturing Systems

In this approach one or more components of traditional manufacturing systems are turned into or enriched with intelligent systems. As shown in Figure 1.18, all other components of the system work in the traditional manner and AI support is provided only in one or more of those.

An example to this approach can be intelligent scheduling systems where job schedules are created using AI techniques and all other manufacturing activities are carried out in traditional ways. There are many scheduling algorithms. Problem solvers define their problem and select one of the most convenient algorithms for their own plant. When the number of jobs and machines increases, the selection of the problem becomes very complex. The decision will require a huge amount of knowledge to be taken into account. Sometimes this knowledge may not be available in the time needed. Since the knowledge base of an intelligent system is populated beforehand, the computer may take all aspects of the manufacturing systems into account and create implementable schedules for certain manufacturing systems.

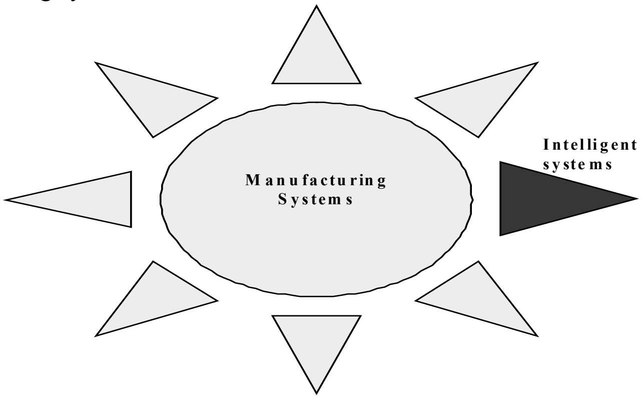


Fig. 1.18 An AI-supported manufacturing system

1.4.2.2 Artificial-intelligence-integrated Manufacturing Systems

In this approach, AI and manufacturing systems are independent of each other but they can provide support to each other (see Figure 1.19). In manufacturing systems, some activities such as creating design for a particular product may be outsourced. Creating rapid prototyping and modelling for manufacturing systems is another example. Intelligent systems and manufacturing systems may not be operating in the same location but need to be integrated for the sake of creating products in accordance with the objectives of the manufacturing systems.

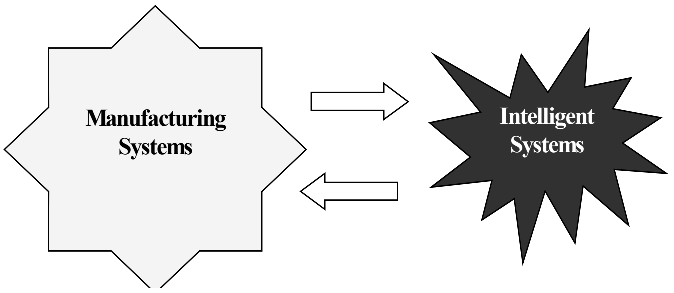


Fig. 1.19 AI-integrated manufacturing systems

1.4.2.3 Totally Intelligent Manufacturing Systems

In this approach, AI is in the heart of manufacturing systems. It keeps the overall control of manufacturing functions and interacts with the operators wherever possible or needed. The basic characteristic of these systems is the ability to explain their own decisions. Figure 1.20 shows the components of these kinds of systems.

Unmanned manufacturing systems are good examples of this approach. In this case, manufacturing activities are totally handled by the robots and they can communicate, cooperate and coordinate their actions together. Manufacturing activities such as path planning for the robots, process plans to work, production schedules to be followed by the machines, etc. are all carried out with the help of intelligent systems. These systems are also capable of correcting themselves in case of a failure or fault. They can utilize all kinds of knowledge, models and data available in the knowledge base, model base and databases respectively.

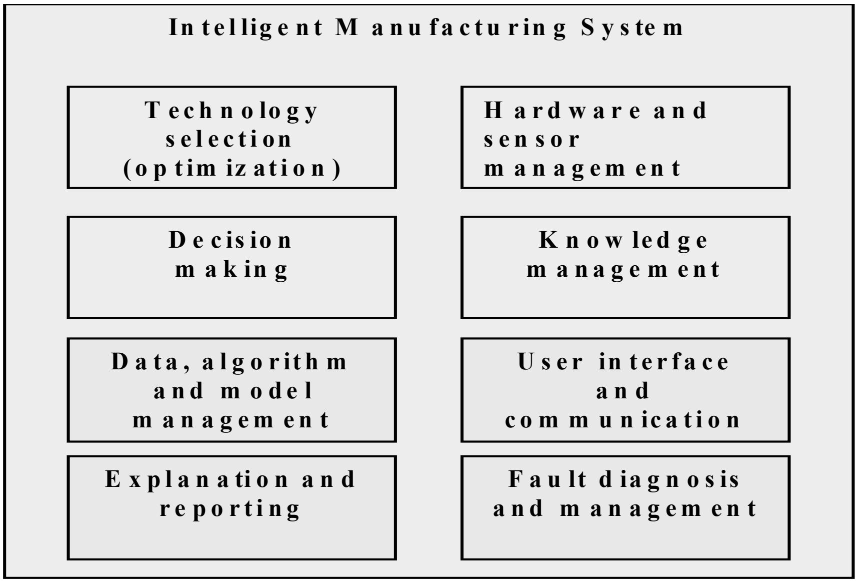

Intelligent Manufacturing System
Technology selection (optimization)
Hardware and sensor management
Decision making
Knowledge management
Data, algorithm and model management
User interface and communication
Explanation and reporting
Fault diagnosis and management

Fig. 1.20 Components of total intelligent manufacturing systems

1.5 Properties of Intelligent Manufacturing Systems

Intelligent manufacturing systems should have the following characteristics (see Rzevski, 1997).

- *Adaptation* is one of the most important features of intelligent manufacturing systems. They should be able to adapt themselves to the environment in which they are operating without compromising the manufacturing objectives.
- *Automated maintenance* is also an important aspect of intelligent manufacturing systems. Systems should be able to identify the failures and take corrective

actions without human intervention (wherever possible). They could be reconfigurable in this sense.

- *Communication* cannot be disregarded in intelligent manufacturing systems, as it is the only way to cooperate among the manufacturing components. The system may produce reports, direct orders to some components and cooperate together with another system to perform certain activities, etc.
- *Autonomy* is the level of independency of intelligent manufacturing systems without which performance of intelligent behaviour could be limited.
- *Learning* is the capability of the systems to improve their knowledge on the basis of the knowledge available in the knowledge bases. This capability seems to be one of the indispensable features of intelligent manufacturing systems.
- *Self-progress* indicates the capability of intelligent manufacturing systems to evolve in time. This can be achieved either by learning or updating the knowledge in the knowledge bases. This can also be triggered by experimenting with the existing knowledge and evaluating the respective performance.
- *Estimation* capability makes intelligent manufacturing systems able to foresee the changes and their possible effects on manufacturing systems.
- *Goal seeking* is the capability of the systems to create goals, refine or update the existing ones in accordance with the progress and mission of the system. In order to achieve a predefined goal, it may be decomposed into smaller ones, which could be managed more easily.
- *Creativity* is also an emerging characteristic of intelligent manufacturing systems. These systems are expected to create new working principles, new theories, forecasts, etc. This capability necessitates interaction with human and other manufacturing components as well as a high degree of autonomy.
- *Reproduction* is to produce or to reuse system component whenever needed. Intelligent manufacturing systems may provide facilities for the same system to be utilized in different places at the same time.

Similarly, Madejski (2008) provided the following properties to become the dominant features of intelligent agents, which establish the baseline for intelligent manufacturing environments.

- *Autonomy*: operating without the direct intervention of humans or others, and having control over their actions and internal state.
- *Social ability*: interacting with other agents (and possibly humans) through an agent-communication language.
- *Reactivity*: perceiving the environment (which may be the physical world, a user via a graphical user interface, a collection of other agents, the internet, or perhaps all of these combined), and responding in a timely fashion to changes that occur in it.
- *Pro-activeness*: exhibiting a goal-directed behaviour by taking the initiative.

1.6 Architecture of Intelligent Manufacturing Systems

As in all kinds of intelligent systems, intelligent manufacturing systems are to be equipped with a self-decision-making capability. In addition to traditional manufacturing functions, they possess knowledge bases and respective software to utilize the existing knowledge. There have been various approaches to designing intelligent manufacturing systems. Among them is the reference model for intelligent manufacturing systems (REMIMS) developed by Tekez (2007) under the supervision of the author of this chapter. As reported in Oztemel and Tekez (2009a), REMIMS is designed to provide a general framework for dynamic integration and standard knowledge exchange infrastructure in distributed environments. It requires a manufacturing enterprise to be organized into management units (modules) each of which consist of a group of intelligent agents. Each of these agents possesses a particular combination of knowledge, skills, and abilities. Taking these into account, REMIMS is reviewed here in three different aspects, namely:

- management of processes and workload (*system view*);
- knowledge and information flow (*knowledge view*); and
- logical inferences and intelligent capabilities (*reasoning view*).

Figure 1.21 shows the relationship of these different views.

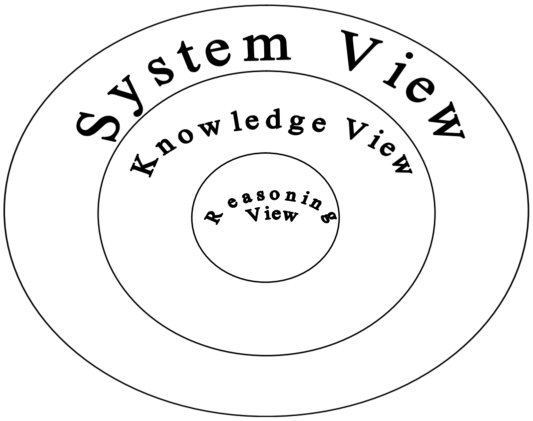


Fig. 1.21 Different aspects of REMIMS (Oztemel and Tekez, 2009a)

System view includes the description of the overall manufacturing systems and concentrates on system functionalities in order to set up smooth interrelationships. As indicated in Table 1.1, REMIMS includes mainly the following manufacturing systems: marketing, design, planning, supply chain management, manufacturing, quality, material management, research and development, and performance monitoring. Also note that REMIMS has a hierarchically nested architecture with four layers each of which is coded as indicated. Although four levels are taken into consideration, the structure of REMIMS allows for the systems to be extended both vertically and horizontally. In this regard, REMIMS offers the flexibility of adding new manufacturing agents, which will support improvement and customization efforts to new requirements if any.

Table 1.1 Main functional components of REMIMS (Oztemel and Tekez, 2009a)

Manufacturing functions			
First level	**Second level**	**Third level**	**Fourth level**
01. Marketing	01.01. Forecasting		
	01.02. Customer relation management	01.02.01. Sales management	
		01.02.02. Customer information system	
02. Design	02.01. Existing product improvement	02.01.01. Prototype production	02.01.01.01. Functional characteristics
			02.01.01.02. Geometric modeling
	02.02. New product design	02.02.01. Generation of bill of material	
03. Planning	03.01. Production planning	03.01.01. Master plan generation	03.01.01.01. Material requirement planning
			03.01.01.02. Capacity planning
		03.01.02. Production plan generation	03.01.02.01. Scheduling
	03.02. Process planning	03.02.01. Process routing	
		03.02.02. Equipment selection	
		03.02.03. Facility layout	
04. Supply chain management	04.01. Purchasing	04.01.01. Supply	
		04.01.02. Material transportation	
	04.02. Supplier evaluation		

Table 1.1 (continued)

Manufacturing functions			
First level	**Second level**	**Third level**	**Fourth level**
05. Manufacturing	05.01. Work flow management	05.01.01. Shop floor management	
		05.01.02. Assembly management	
		05.01.03. Maintenance management	
	05.02. Monitoring and control		
06. Quality	06.01. Quality management systems	06.01.01. Incoming inspection	
		06.01.02. Manufacturing inspection	
		06.01.03. Assembly inspection	
		06.01.04. Product audit	
		06.01.05. After sales	
	06.02. Quality assurance		
07. Material management	07.01. Storage management	07.01.01. Raw material management	
		07.01.02 Product management	
	07.02. Order management		
08. Research and development			***Horizontal extension (additions)*** →
09. Performance monitoring		***Vertical extension (additions)*** ↓	

In addition to hierarchically distributed agents, REMIMS is enriched with a knowledge form base, external resource base and database management systems. Within the proposed framework, the agents are supposed to communicate their knowledge and even negotiate with each other through a so-called *knowledge network* (*KN*) over a distributed manufacturing environment (see Figure 1.22). It should be noted that the proposed reference model presents an ideal manufacturing environment from which the users or developers can establish their manufacturing systems through a tailoring process based on their needs and particular manufacturing requirements.

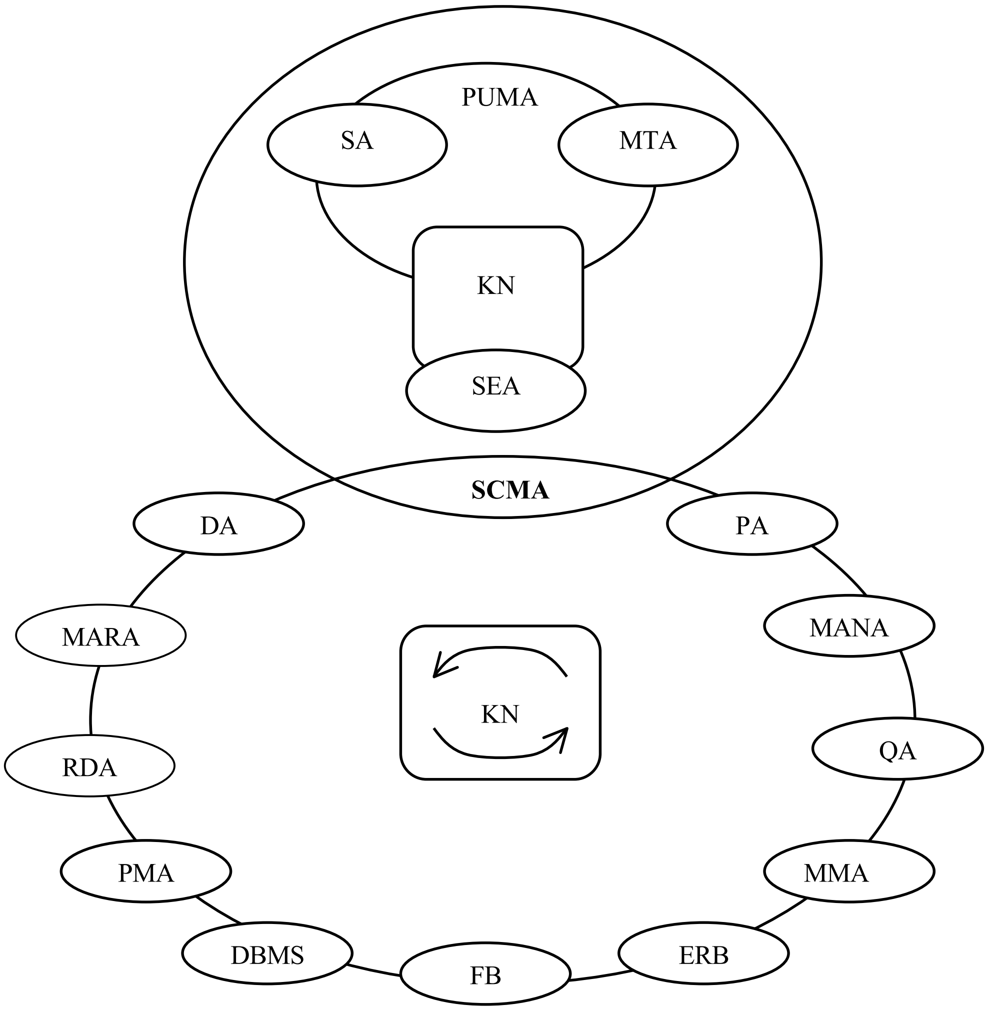


Fig. 1.22 Nested architecture of REMIMS (Oztemel and Tekez, 2009a). *KN: knowledge network, DA: design agent, MARA: marketing agent, RDA: research and development agent, PMA: performance monitoring agent, DBMS: database management system, FB: form base, ERB: external resource base, MMA: material management agent, MANA: manufacturing agent, QA: quality agent, PA: planning agent, SCMA: supply chain management agent, SA: supply agent, MTA: material transportation agent, SEA: supplier evaluation agent, PUMA: purchasing agent*

Knowledge view points out the capabilities of knowledge and information flow among the manufacturing agents. Agents who are equipped with certain manufacturing knowledge should be able to perform their actions as effectively as possible and communicate the results to other agents who may need them. REMIMS provides a certain standard for knowledge exchange among the agents. This is done through a KN as indicated in Figure 1.22. It should be noted that an agent might produce certain knowledge, which would be a strict requirement by some other agents. Similarly it may need some inputs, which could be produced by other agents within the integration framework (see Figure 1.23, for possible information exchanged). A standard knowledge exchange procedure is therefore extremely useful in order to handle automated exchange of information as well as knowledge. In order to facilitate this, REMIMS provides a standard knowledge exchange scheme, which may be called a "knowledge protocol". The KN therefore can be considered as a distributed information system providing ways of agent interactions by several means, in this case, "knowledge protocols" and "knowledge forms" which are explained in detail elsewhere (see Oztemel and Tekez, 2009b, 2009c).

Reasoning view highlights the basic capabilities of decision making, problem solving and reasoning capabilities of each agent responsible for carrying out a certain manufacturing function. REMIMS is composed of a set of intelligent agents each of which is responsible for performing a different manufacturing activity. Each agent may in turn be composed of several sub-agents. The logical view of the reference model is task dependent and can have different architectures for different tasks. Each agent within the reference model can have a general architecture as shown in Figure 1.24.

In order to perceive and act (reason) about the events, the agents should be equipped with suitable sensors and effectors as well as with the domain knowledge. In other words, the agent should perform whatever action is expected from it to maximize its performance. The actions should be decided on the basis of the evidence provided by the percept sequence and whatever "built-in knowledge" the agent possesses. As seen in Figure 1.24, an agent may be composed of:

1. *situation assessment module* to perceive the events from the environment;
2. *reasoning and decision-making module* to provide solutions to perceived information;
3. *behaviour representation module* to act according to the response produced by the reasoning and decision module.

By being agent-based architecture, REMIMS emphasizes the importance of knowledge because of interdependencies among the agents. Agents interact with each other and coordinate their actions derived from their knowledge. They have to utilize their knowledge in an integrated fashion in order to link the respective tasks carried out by different agents. Each agent can work independently utilizing the required knowledge, which may even have been produced by other agents.

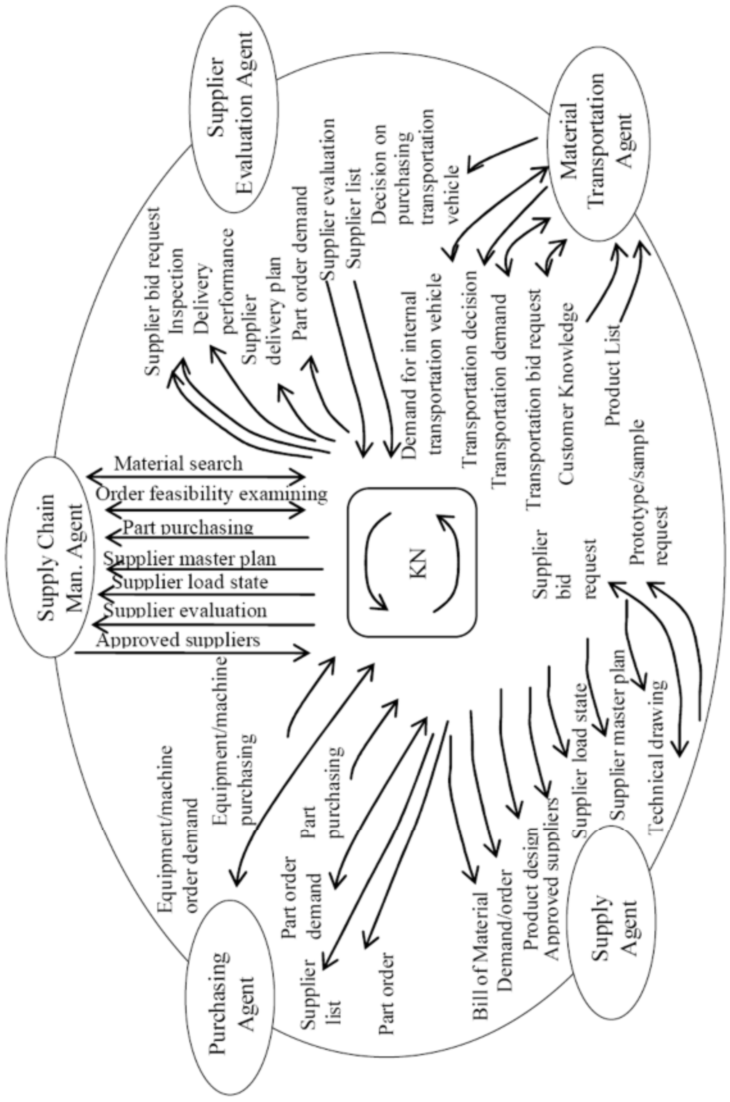


Fig. 1.23 An example of knowledge exchanges on knowledge network of REMIMS (Oztemel and Tekez, 2009a)

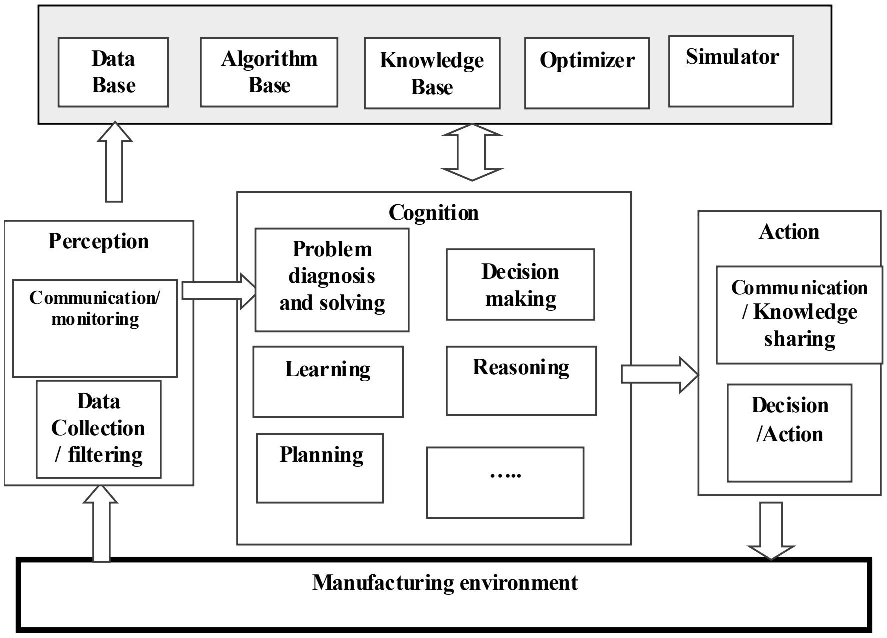


Fig. 1.24 General architecture of a REMIMS agent (adapted from Oztemel and Tekez, 2009a)

1.7 Holonic Manufacturing Systems

Progress in intelligent manufacturing systems promotes the concept of holonic manufacturing which establishes a baseline for autonomous, highly flexible, agile, reusable and modular manufacturing units. That is to say that these systems are designed through autonomous, cooperative, intelligent modules capable of reconfiguring manufacturing systems automatically in response to new system requirements or environmental changes. These systems are constructed using some autonomous structures so called “holons”. Holons are capable of working under the control of others as well as independent of others. That means if one manufacturing unit (holon) fails or cannot respond to some problems; other holons create a reroute of operations in order to avoid major disruptions. It is quite similar to the nature of an ant or termite colony. Holonic manufacturing system is believed to make an important breakthrough in the field of decentralized control for intelligent manufacturing systems. It is important to take note of the two aspects of holonic manufacturing which are:

- event-driven real-time control strategies; and
- deliberative non- real-time distributed information processing.

These two characteristics make holonic manufacturing systems capable of altering machine configurations and production schedules in accordance with immediate and imminent requirements. This is important as it allows the manipulation of breakdowns of the machines and real-time re-scheduling. That makes the manufacturing system agile enough to cope with unexpected changes.

The components of holonic manufacturing (so-called building blocks) are designed to reflect the fact that different manufacturing units behave in an autonomous and cooperative manner. The generic principles underpinning holonic systems were first proposed by Koestler (1967). The concept is being reiterated by the researcher with respect to intelligent manufacturing through illustrating each manufacturing unit as different holons which may utilize different type of knowledge.

Bussmann and McFarlane (1999) identified the basic architecture of a holon as shown in Figure 1.25 and highlighted the key properties of holonic manufacturing system as the following:

- autonomy;
- cooperation;
- self-organization; and
- reconfigurability.

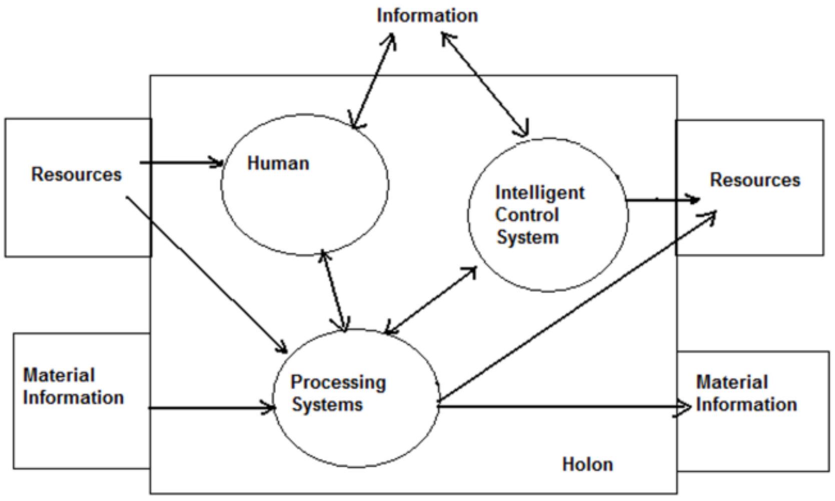


Fig. 1.25 A basic architecture of a holon (Bussmann and McFarlane, 1999)

There are mainly two types of holons in a holonic manufacturing system. They are:

- *Resource holons*, which can provide all the generic resources in the intelligent manufacturing system. Each of these resource holons is an entity that performs an action over an item. They include painting machines, automated guided vehicles (AGVs), inspection stations with video cameras and AI software to recognize any defects in the items, etc.

- *Product holons*, which represent the requirements of operations such as production, assembly and so forth. These holons also provide knowledge on how to achieve the manufacturing objectives. They can provide expert advice, and may also act as an information server to disseminate knowledge among the holons. Each can be re-used in the scope of different operations and each could negotiate with various resource holons in order to secure the desired services. In other words, each product holon is an active entity responsible for performing the manufacturing management work correctly and on time, while explicitly capturing all information processing needed for a specific job.

Holons establish a holarchy which is a system of holons that can cooperate to achieve a certain manufacturing goal or objective. Sigumura *et al.* (1997) indicated the basic structure of a holonic manufacturing system as shown in Figure 1.26.

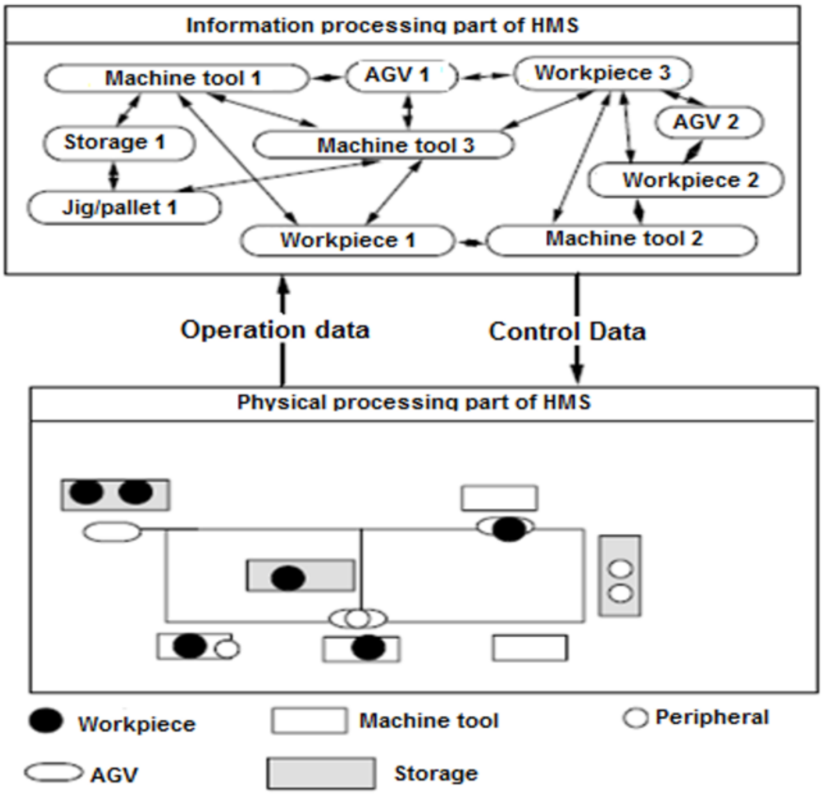


Fig. 1.26 Basic structure of holonic manufacturing system (Sigumura *et al.*, 1997)

As indicated in the Figure 1.26, the system is basically divided into the physical processing part and the information processing part, and both parts consist of a set of holonic components. The physical processing part transforms the blank materials to the final products through the autonomous and cooperative activities of the holonic components. The data required in the physical processes are generated and determined in the information processing part, which also consists of a set of holonic components.

Similarly, Sigumura *et al.* (1997) showed the holarchy of holonic manufacturing systems as illustrated in Figure 1.27.

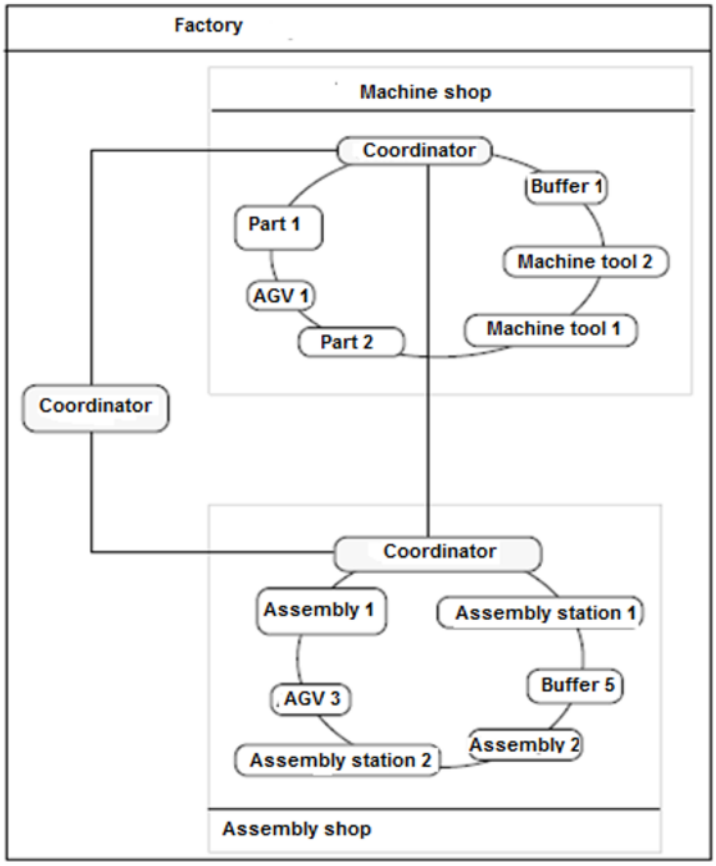


Fig. 1.27 Holarchy of holonic manufacturing systems (Sigumura *et al.*, 1997)

This figure shows an example of the holarchy in holonic manufacturing systems consisting of a machine shop and an assembly shop. The individual shops include such holonic components as machine tools, assembly stations, parts and assemblies. In the holarchy, local-level decisions, such as shop scheduling, are made through the autonomous decisions of holonic components, and their cooperation. The coordinators in the shops coordinate and/or constrain the autonomous activities of individual holonic components. Some more detailed information on holonic manufacturing systems can be found in Jarvis *et al.* (2008).

1.8 Applications of Intelligent Manufacturing Systems

There have been numerous examples of intelligent manufacturing in the last two decades. These are presented in many conferences and papers and can be found in fielded applications in industry. It is not possible to review all of these applications here in this book. Several examples will be provided in order to highlight possible areas and provide some hints on practical use of these systems.

Pham and Oztemel (1996) reported their work on intelligent quality systems in their book. They have reported an expert system called XPC capable of performing statistical process control. XPC, which was developed by Oztemel (1992) constructs control charts automatically, and performs process capability analysis in order to make sure that the process is designed in such a way that it may satisfy customer requirements within tolerances. It is capable of performing on-line monitoring of the process and identifies any faults as well as recommending corrective actions. It updates the chart according to improvements achieved.

Kusiak (1990) wrote a book on intelligent manufacturing systems and reported various intelligent systems from design to equipment selection. Similarly, Norrie et al. (1990) recommended an integrated manufacturing management planning system, which takes process planning, group technology, scheduling, and simulation into account. They created a decision support system called FLEXES utilizing knowledge bases and simulation systems for the sake of efficient and effective manufacturing.

Scherer and Brown (1995) introduced intelligent scheduling in their book. They provided a baseline for the future applications. For example, Monfared and Yang (2005) developed a multi-level scheduling and control algorithm based on the proposed methodologies.

AI is highly utilized in design activities as well. Preparation of manufacturing designs using CAD tools and support of CIM systems have been attracting the developers of CAD. Abouel Nasr and Kamrani (2008) provided an extensive analysis on this issue. Integrating AI and Internet technology for the success of manufacturing design is very popular nowadays. A good example of this is the injection moulding design system developed by Mok *et al.* (2008). In this study, a Java-based intelligent design system equipped with AI technologies is introduced. It is reported that this system increased the design speed and supported the creation of design standards. Similarly, Balic *et al.* (2006) developed an intelligent system for computer numerical-controlled tuning operations using genetic algorithms. Information on other design-related intelligent systems can be found in Jedrzejewski (2007).

Similarly to design systems, intelligent process planning-systems have also been developed. Especially researchers such as Wong (1993) led the studies and provided baselines for the development activities. Main motivations of these systems are reported to be automatic information processing and defining process plans according to design specifications. Mo and Woodman (2005) explained a framework to prepare process plans using the Internet. Deb *et al.* (2006) presented a neural-network-based process-planning system. The research along this line continues with an increasing momentum.

Intelligent manufacturing systems are emerging in hardware-related studies as well. There have been fully automated and intelligent manufacturing systems and manufacturing robots developed and effectively used in manufacturing shops. Balic *et al.* (2003) assessed the work along this line in their book and provided useful information in detail. Tsourveloudis *et al.* (2000) reported a neural network and fuzzy-logic-based robot gripping system. Similarly, Ioannidis *et al.* (2004) introduced fuzzy-logic controllers. A very informative review on robotics technology is also provided by Jarvis (2008).

Attention has also been given to the integration of intelligent manufacturing systems. Agent-based systems are good examples of such applications. There have been countless examples of applications of agents in manufacturing. To mention some, Zhou *et al.* (1999) provided a real-time monitoring system, which could be utilized by agents in an integrated environment. Shen *et al.* (2007) proposed a service-based integrated architecture for agent cooperation, which could be used in manufacturing systems. Tekez (2007) created a general agent-based reference model for integrated intelligent manufacturing systems, naimely (REMIMS) as reported above.

The progress in intelligent manufacturing systems also directed the attention of the researchers towards distributed holonic systems. Aguayo *et al.* (2008) presented a protomodel called FRABIHO which constitutes distributed and intelligent engineering production systems, derived from the synthesis of the models proposed in the frame of the fractal, bionic and holonic intelligent manufacturing systems.

Similarly, Hou and Gong (2008) introduced a knowledge centric intelligent manufacturing system (KCIMS), which is devised to assist in tackling the current challenges posed to the semiconductor manufacturing services industry by the competitive environment and pressure of demanding customer. However, the approach proposed could be adapted to other industrial sectors as well. The overall KCIMS framework can be mainly classified into three parts, namely knowledge management (KM) basic structure, action-doer structure and advance product quality planning process (APQP) based structure. The first part includes the most important structure of control, applications and interface. The second part deals with the repository services. The framework utilizes intelligent agents to facilitate the tasks of repository services at the third part. A modified APQP, called a intelligent quality planning process (IXQP, where X stands for process, material, human resources, efficiency and technology) breaks down the APQP process in small subcomponents so as to link it through a KM-based system and have better control on company supply chain operations.

1.9 Conclusions

Intelligent manufacturing systems are evolving with an increasing speed. New systems are being developed in very short time frames. It is now apparent that intelligent manufacturing systems will be more dominant in industrial and manufacturing areas and:

- They will make it possible to reduce the size of the product but increase their functionality as well as complexity.
- They will be equipped with highly effective sensors for perceiving inputs and acting accordingly.
- They will be capable of collecting/filtering and reasoning about the knowledge and data without any extra burden on the systems.
- They will reconfigure themselves in the light of the progress in manufacturing systems and technologies.
- They will have an extensive capability for learning and planning as well as updating their knowledge accordingly.
- They will be able to adapt themselves to any kind of manufacturing environment.
- They will be able to define the best manufacturing methods for their intended purposes.
- They will increase their performance continuously and reduce the human involvement.

It is now obvious that the next decade will be the decade of unmanned factories and new changes will continue to surprise both the academic and the industrial community.

References

Abouel Nasr ES, Kamrani AK (2008) Intelligent design and manufacturing. In: Collaborative engineering, Chapter 6, Springer, ISBN 978-0-387-47319-2

Aguayo F, Lama JR, Carrilero MS et al. (2008) Intelligent agents based manufacturing systems: A formal specification derived from FRABIHO. In: Pham DT, Eldukhri EE and Soroka AJ (Eds.) Innovative production machines and systems, pp. 43–48, ISBN 978-1904445-52-4

Balic J, Valavanis KP, Tsourveloudis NC et al. (2003) Intelligent manufacturing systems: Programming and control. University of Maribor Publications, ISBN 86-435-0546-3

Balic J, Kovacic M, Vaupotic B (2006) Intelligent programming of CNC turning operations using genetic algorithm. J. Intell. Manuf., 17(3):331–340

Bamber L, Dale BG (2000) Lean production: a study of application in a traditional manufacturing environment. Prod. Plan. Control, 11(3):291–298

Bosque M (2002) Understanding 99% of artificial neural networks: Introduction and tricks. Writers Club Press, USA, ISBN 0-595-21996-9

Brun A, Portioli, A (1999) Agent-based shop floor scheduling of multi stage systems. Comput. and Ind. Eng., 37(1):457-460

Bussmann S, McFarlane DC (1999) Rationales for holonic manufacturing control. In: Brussel H van and Valckenaers P (Eds.), Proceedings of the 2nd International Workshop on Intelligent Manufacturing Systems, pp. 177–184

Cantamessa M (1997) Agent-based modelling and management of manufacturing systems. Comput. In Ind., 34:173-186

Chryssolouris G (2006) Manufacturing systems: theory and practice, Mech. Eng. Series, Springer-Verlag 2nd ed., ISBN 978-0-387-25683-2

Cox E (1999) The fuzzy systems handbook. 2nd Ed., Academic Press, ISBN 0-12-194270-8

Deb S, Ghosh K, Paul S (2006) A neural network based methodology for machining operations selection in computer-aided process planning for rotationally symmetrical parts. J. Intell. Manuf., 17:557–569

Hannam, R (1997) Computer integrated manufacturing: From concepts to realization. Addison Wesley Longman, ISBN 978-0-201-17546-2

Hou TC, Gong DC (2008) Knowledge management centric intelligent manufacturing systems for semiconductor manufacturing services industry. J. Chinese Inst. Ind. Eng., 25(6):510–518

Ioannidis S, Tsourveloudis NC, Valavanis KP (2004) Fuzzy supervisory control of manufacturing systems. IEEE Trans. Rob. Auto., 20(3):379–389

Jarvis R (2008) Intelligent robotics: past, present and future. Int. J. Comput. Sci. Appl., 5(3):23–35

Jarvis J, Jarvis D, Ronnquist R et al. (2008) Holonic execution: A BDI approach, Springer-Verlag, Berlin Heidelberg, ISBN 978-3-540-77478-5

Jedrzejewski J (2007) Manufacturing intelligent design and optimization processes. Editorial Institution of the Wroclaw Board of Scientific Technical Societies Federation, Poland, ISSN 1895-7595

Kusiak A (1990) Intelligent manufacturing systems. Prentice Hall International Editions, New Jersey, ISBN 978-0-134-68364-5

Kusiak A (2000) Computational intelligence in design and manufacturing. John Wiley and Sons, ISBN 0-471-34879-1

Koestler A (1967) The ghost in the machine, Arkana, ISBN 9780140191929

Madesjki J (2008) Agent architecture for intelligent manufacturing systems. J. Ach. Mat. Manuf. Eng., 29(2):167–170

Meystel A, Albus JS (2001) Intelligent systems: Architecture, design, and control. Wiley Series on Intelligent Systems, John Wiley and Sons, Inc., New York, ISBN 0-471-19374-7

Mo JPT, Woodman S (2005) Intelligent process planning and control framework for the internet. Knowledge and skill chains in engineering and manufacturing, IFIP International Federation for Information Processing, Vol. 168. ISBN 978-0-387-23851-7

Mok CK, Chin KS, ve Lan H (2008) An internet-based intelligent design system for injection moulds. Rob. Comput. Integr. Manuf., 24(1):1–15

Monfared MAS, Yang JB (2005) Multi-level intelligent scheduling and control system for an automated flow shop manufacturing environment. Int. J. Prod. Res., 43(1):147–168

Norrie, DH, Fauvel OR, Gaines BR (1990) Object-oriented management planning systems for advanced manufacturing. Int. J. Comput. Integr. Manuf., 3(6):373–378

Oztemel E (1992) Integrating expert systems and neural networks for intelligent statistical process control. Unpublished Ph.D thesis, University of Wales, College of Cardiff

Oztemel E, Tekez EK (2009a) A general framework of a reference model for intelligent integrated manufacturing systems. Eng. Appl. Artif. Intell. (Int. J.), 22(6):855–864

Oztemel E, Tekez EK (2009b) Knowledge protocol. In: Cunha MM, Oliveira EF, Tavares AJ, Ferreira LG (Eds) Handbook of research on social dimensions of semantic technologies and web services, Chapter 11, IGI Global Publishing, ISBN 9781605666501

Oztemel E, Tekez EK (2009c) Integrating manufacturing systems through knowledge exchange protocols within an agent based knowledge network. Rob. Comput. Integr. Manuf., 25:235–245

Pham DT, Oztemel E (1996) Intelligent quality systems. Springer-Verlag, London, ISBN:3-450-76045-8

Rabunal JYR, Dorado J (2006) Artificial neural networks in real-life applications. Idea Group Publishing, ISBN 1-59140-902-0

Revees CR, Rowe JE (2004) Genetic algorithms–principles and perspectives. 2nd Ed, Kluwer Academic Press, ISBN 1-4020-7240-6

Russel S, Norvig P (2002) Artificial intelligence: A modern approach. 2nd Ed., Prentice Hall International, New Jersey, ISBN 0-13-790395-2

Rzevski GA (1997) Framework for designing intelligent manufacturing systems. Comput. In Ind., 34:211–219

Sanchez LM, Nagi R (2001) A review of agile manufacturing systems. Int. J. Prod. Res., 39(16): 3561–3600

Scherer WT, Brown DE (1995) Intelligent scheduling. Kluwer Academic Publishers, Norwell, MA, ISBN 0792395158

Shen W, Norrie DH (1999a) Developing intelligent manufacturing systems using collaborative agents. In : Proceedings of the 2nd International Workshop on Intelligent Manufacturing Systems (IMS), H Van Brussel and P Valckenaers (Eds.), Katholieke Universiteit Leuven, ISBN 90-73802-69-5 pp. 157–166

Shen W, Norrie DH (1999b) Agent-based systems for intelligent manufacturing: A state of the art survey. Know. Info. Syst. Int. J., 1(2):129–156

Shen W, Hao Q, Wang S et al. (2007) An agent-based service-oriented integration architecture for collaborative intelligent manufacturing Rob. Comput. Integr. Manuf., 23(3):315–325

Sugimura N, Tanimizu Y, Yoshioka T (1997) A study on object oriented modelling of holonic manufacturing system. In: Proceedings of the 29th CIPR International Seminar on Manufacturing Systems, May 13–14

Smutny L (2003) Intelligent measurement, diagnostic and control systems. Acta Montanistica Slovaca Ročník 8

Tekez E K (2007) A reference model for integrated intelligent manufacturing system. Unpublished PhD thesis, Sakarya University, Turkey

Tsourveloudis N, Kolluru R, Valavanis KP et al. (2000) Suction control of a robotic gripper: A neuro-fuzzy approach. J. Intell. Rob. Syst., 27(3):215–235

Turban E, Aronson JE, Liang TP (2008) Decision support systems and intelligent systems. Prentice Hall International Editions, 7th Ed, ISBN 978-0131230132

Wong TN (1993) Development of a knowledge-based automated process planning system. Ind. Eng., 1992–1993, pp. 44–48

Wooldridge M (2009) An introduction to multi-agent systems. 2nd Ed. John Wiley and Sons Ltd., England, ISBN 9780470519462

Zhou B, Wang L, Norrie DH (1999) Design of distributed real-time control agents for intelligent manufacturing systems. In: Proceedings of 2nd International Workshop on Intelligent Manufacturing Systems, Leuven, Belgium, pp. 237–244

Chapter 2
Agent-based System for Knowledge Acquisition and Management Within a Networked Enterprise

A.J. Soroka

Abstract The examination of tasks involved in the gathering and processing of fault information with enterprises has shown that there are several stages where human errors can occur and has also revealed inefficient and time-consuming operations, resulting in bottlenecks that can reduce the potential benefits of automatic rule generation. Therefore, these various tasks could themselves be automated through the use of agent-based systems and machine learning techniques. This chapter shows that it is possible to automate the gathering and manipulation of fault reports using an agent-based system. Such a system can remove the need for manual processing of fault reports and the problems that may result from this. It also shows that a finite state automata (FSA)-based agent architecture is suitable for application in this particular problem domain, due to the reactive nature of an FSA. An FSA and state-table approach coupled with the modularity of the system should also enable it to be modified readily for different applications.

Keywords Multi-based system, machine learning, knowledge acquisition, networked enterprise

2.1 Agent-based and Related Systems

"Agent" has become a very popular term in the fields of computer science and artificial intelligence (AI) (Nwana and Ndumu, 1996). Agent-based systems are being heralded as the next significant breakthrough in software development and the new revolution in software (Wooldridge and Jennings, 1995). In comparison to other topics of AI research, agent research is comparatively young. There is no unanimously agreed date as to when agent research began, for example in Maes

A.J. Soroka (✉)
Manufacturing Engineering Centre, Cardiff University, Queen's Buildings, Cardiff CF24 3AA, UK
e-mail: SorokaAJ@cardiff.ac.uk

(1995) 1985 is cited, yet in Wooldridge and Jennings (1995) and Nwana and Ndumu (1996) 1977 is cited. Therefore, it could be considered that the majority view is that the start date is 1977, pertaining to Hewitts work on actors (Hewitt, 1977). However, the term intelligent or autonomous agent is not explicitly mentioned in Hewitt (1977) and this is the reason for some of the confusion relating to when agent research began.

2.1.1 Origins of Agent Research

Agent research as it currently stands can be considered to have had perhaps two inter-related starting points, namely blackboard (BB) systems and distributed artificial intelligence (DAI). Therefore these two areas are briefly discussed before agents are considered in more detail.

Blackboard Systems

The BB system approach to problem-solving is based upon the concept of several knowledge sources (KSs) working on a problem via a globally accessible data structure known as a BB, which contains all solution elements generated during problem solving (Englemore and Morgan, 1988; Craig, 1995). The only communication these KSs have between one another is by adding or modifying entries on a BB. The operation of the KSs is normally governed by some form of control system, known as a scheduler, which implements the chosen problem-solving strategy (Craig, 1995). The KSs within a BB system are essentially expert systems that are dedicated to solving problems in a specific area. A typical structure of a BB system is shown in (Figure 2.1). Even though the basic ideas of BB and agent systems are similar, they differ from each other as it can be considered that there is no autonomy within a BB system. This is because of the scheduler, which reviews the KSs and selects the most appropriate KS. Therefore, the KSs do not decide of their own accord whether to act or not.

BB systems have typically been applied to tasks such as speech understanding (Hearsay II (Erman *et al.*, 1988)) and image understanding (Dreper *et al.*, 1988), where multiple KSs are required to perform the overall task. The technique of using a common BB is still being employed today, along with other related technologies, but in the domain of agent-based systems where agents replace KSs. Several examples exist of such applications. One is Guardian (Hayes-Roth *et al.*, 1994), which is an agent-based patient-monitoring system for an intensive care unit that employs a BB-based control architecture. Another example is described in Holt and Rodd (1994), which discusses how a dynamic BB is used to facilitate inter-agent communication within a manufacturing system. Concepts from BB systems have also been applied in the area of DAI and agent systems for conflict resolution and co-operative working within an intelligent design system (Lander *et al.*, 1990).

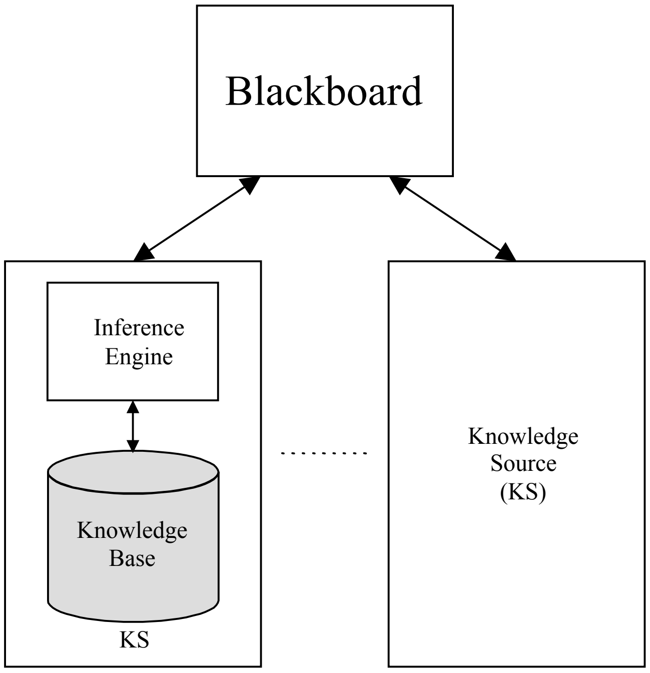


Fig. 2.1 Structure of a typical BB system

Distributed Artificial Intelligence

Distributed AI is based upon the principle of co-operative problem solving by a decentralised group of agents. These agents can take the form of a simple processing element or a complex entity (intelligent agents) (Huhns, 1987). The term distributed is used because in DAI the processing elements (or agents) are distributed amongst many computers, thereby creating a problem-solving system that is very robust.

DAI research can be split into three different areas, distributed problem solving, multi-agent systems (MAS), and parallel AI (Bond and Gasser, 1988). The MAS research conducted within the area of DAI tended to concentrate upon *macro* phenomena (the social level) (Wooldridge and Jennings, 1995). This resulted in a great deal of work having been carried out on collaborative and co-operative agents and their interaction for joint problem solving. Examples of research conducted in this area include work by Georgeff concentrating on methodical and theoretical aspects (Georgeff, 1988). A significant amount of work was performed on inter-agent negotiation and collaboration using various approaches

(Sycara, 1989; Jennings, 1992, 1995; Osawa and Tokoro, 1992; Khedro and Genesereth, 1994; Sen *et al.*, 1994). Some research in this area looked specifically at applying game-theory techniques (Axelrod, 1984) to multi-agent systems (Zlotkin and Rosenschein, 1989; Ito and Yano, 1995), in an attempt to foster and facilitate co-operation between agents. From this work relating to co-operative behaviour, several frameworks and systems were proposed for the collaborative operation of agent systems (Lander *et al.*, 1990; Jennings, 1992, 1995; Jennings *et al.*, 1992).

A natural extension of the work pertaining to agent collaboration and co-operation can be considered to be the creation of agent communication and knowledge interchange languages and formats. Examples include the Knowledge Interchange Format, which deals with message content (Genesereth and Fikes, 1992), Knowledge Query and Manipulation Language, which is concerned with the performance of actions (Mayfield *et al.*, 1996; Labrou and Finin, 1997) and also the FIPA (Foundation for Intelligent Physical Agents) agent communication language specification (FIPA, 1997). Such languages were aimed at enabling communications between homogenous agents operating within a system and also heterogeneous agents existing within different systems created by different people, thus attempting to achieve the goal of interactivity between all agents.

It can perhaps be considered that DAI and agent system research have become synonymous with each other in recent times, with researchers in one domain conducting research in the other. As such there exist agent-based systems whose origins are within the area of DAI.

2.1.2 Definition of an Agent

Currently there exist many different "definitions" of what an agent is and names for "agents", for example, intelligent agents, autonomous agents, robots or hardware agents, and softbots or software robots. This has resulted in a degree of confusion such that the question "What is an agent?" is regarded to be as embarrassing to the agent-based computing community as the question "What is intelligence?" is to the AI community (Wooldridge and Jennings, 1995). In (Wooldridge, 1996ab) it is said that obtaining a definition for an agent is like the Rorschach test, because every person has his own answer. Whilst at first glance this may appear to be a cynical opinion, the next section illustrates that this is very much the case.

This confusion can also be partially attributed to the fact that there were many uses of the term agent prior to its adoption by the agent-based computing community. For example, the term refers to people or organisations that perform services for others such as estate or travel agents. There are also uses in the areas of biology and chemistry, biological and chemical agents respectively, which are either organisms or chemicals having an effect on something else be it animal, mineral or vegetable.

Agent definitions in general can be grouped together according to the sets of characteristics that are specified for an agent. Several definitions specify the characteristics dependent upon the operation of the agents. For instance, agents act within an environment and can sense or receive stimuli from the environment and can act upon and affect that environment to achieve a set of goals (Hayes-Roth, 1993; Gibbins *et al.*, 1994; Maes, 1995; Franklin and Graesser, 1996; Kearney, 1996). Other definitions are based upon features that characterise an agent, such as the ability to act autonomously, react and plan (Jennings and Wooldridge, 1995; Ekdahl *et al.*, 1995; Moffat and Frijda, 1995; Alty, 1997; Wooldridge, 1996ab). Others such as Struthers (1996) define an agent in terms of both structural and functional aspects. Some examples are very specific, such as Genesereth and Ketchpel (1994) and Singh and Huhns (1999) which give the definition that an agent is only an agent if it communicates with another agent.

From this it can be appreciated that it may not necessarily be feasible or practical to give an exact definition of an agent. Indeed it is considered that the only agreement concerning definitions is that there is no one single definition (Singh and Huhns, 1999).

There is, however, a need to give a generic description of ideas that agents encompass. A simple specification is given in Foner and Crabtree (1996). If it were not for the mention of performing personalised tasks, which may or may not be applicable to all agent systems, that description would be generic and encompass the majority of agent definitions. A simple generic definition of an agent adapted from Foner and Crabtree (1996) could be "A piece of software, or hardware, that acts in an autonomous manner to complete a task or tasks". This definition does not specify how an agent operates or what it features but rather can be applied to most agents.

2.1.3 Agent Architectures

Agent architectures are mainly concerned with the implementation of a proposed agent system and the implementation of theories of agency, where the theories deal with issues such as how an agent should behave, its properties, and how to conceptualise these ideas. There are several agent architectures that are commonly utilised: deliberative, reactive, and hybrid.

Deliberative

Deliberative agents are agents that depend upon models of the environment to decide what course of action to take. It is stated in Wooldridge and Jennings (1995) that the term deliberative agent stemmed from Genesereth and Nilsson (1988) who stated that "The key in defining an agent in this class is the use of an automated inference method like resolution in deriving a sentence that indicates the required action on each cycle".

Several examples exist of agents based implicitly on deliberative architectures. A significant proportion of these architectures possess models of beliefs, desires and intentions (BDI) (Bratman *et al.*, 1988; Jennings *et al.*, 1992; Rao and Georgeff, 1995). Others possess similar properties such as beliefs, actions and plans (Kumar and Shapiro, 1991). Beliefs are an expression of what the agent currently knows about its environment, desires are an expression of what the agent wishes the future states of itself or other agents to take and intentions are goals that the agent would like to fulfil. These features enable the agent to make decisions about how to perform a task that are dependent upon BDI such that what an agent will do will be affected by the current state of its model. This attempts to mimic the behaviour of humans, where a decision is typically made based upon the knowledge the person possesses.

Examples of practical applications of deliberative agents include GRATE (generic rules and agent model testbed environment) and its derivations (Jennings *et al.*, 1992, 1993).

However it is considered that deliberative agents, which have to perform theory proving and symbolic manipulation, are hard to implement due to problems with theory proving and the computational complexity of symbolic manipulation (Wooldridge and Jennings, 1995). In Chapman (1987) it was shown that even refined planning techniques such as hierarchical and non-linear planning would prove unusable in time-constrained systems (Wooldridge and Jennings, 1995).

Reactive

The main aim of reactive architectures is to overcome some of the issues and perceived weaknesses of deliberative architectures, thereby creating a robust, flexible and fault-tolerant system (Nwana and Ndumu, 1996).

Reactive agents or agent architectures are also known as reflex, behaviour-based or situated. Reactive agents differ from deliberative agents as they do not possess any models of their environment but rather respond to stimuli. Therefore the agent's reaction depends upon the current state of its environment. In Maes, (1991) three concepts are given that underpin reactive agents: emergent functionality–reactive agents are relatively simple and interact in basic ways and yet complex behavioural patterns can emerge; task decomposition–an agent is viewed as a collection of autonomous modules; reactive agents tend to operate on representations that are close to raw data, in contrast to the high-level symbolic representations that occur in other types of agents.

One potential drawback to reactive systems was considered to be that there was no guarantee that a reactive agent would provide an efficient solution (Knight, 1993). It was, however, discovered that the problem could be solved by increasing the number of reactive agents that are simultaneously attacking a problem. Research has also shown that this is better with regard to improving the efficiency of a reactive system than increasing the amount of deliberative reasoning used.

Most reactive architectures can be considered to be developed around various forms of rule-based architecture (Agre and Chapman, 1987; Kowalski and Sadri, 1996; Meghini, 1997). This is primarily due to agents very much being governed by logical formulas or action rules that specify how they should behave.

It is considered by some that relatively few actual examples of applications of reactive agent architectures exist (Nwana and Ndumu, 1996), and as such there exists no standard mode of operation. However, several can be found, mainly within the robotics or hardware agent domain, which seem to be ideally suited to reactive architectures.

Perhaps the most cited agent system based upon a reactive architecture is PENGI, a game-playing program (Agre and Chapman, 1987). The authors stated that most everyday activities can be considered routine and as such requires little new learning. The PENGI agent decides what to do depending on the current state of the environment.

Other examples include the subsumption and behaviour-based architectures presented in Brooks (1986, 1991). This architecture employs a layer-based control system. These layers control the various behaviours of a mobile robot. In the subsumption architecture these various layers compete with one another to be activated and therefore to control the robot. The lower layers within the system represent relatively primitive operations and can be activated more easily, therefore having precedence over the higher levels. However, these higher levels are able to subsume the roles of the lower ones by suppressing their outputs.

Work similar to that of Brooks has been conducted into agent network architectures (Maes, 1989, 1991). Maes' network comprises types behaviour-based modules or competence modules. These modules will perform certain tasks, but as with the subsumption architecture they must compete to become active. This is because every competence module has an activation level that must be reached before it becomes active. These modules have links connecting them that can be dynamically changed as the system gains more experience, thereby achieving flexible operation.

Another application in the area of robotics is that of situated automata (Kaelbling, 1991; Rosenschein and Kaelbling, 1995). The idea of situated automata is related to that of synchronous sequential digital circuits. These automata are then represented using formal logic. This is then encoded into a program, which is compiled to produce the agent. This approach enables the robot to operate in a time-bounded manner.

Hybrid

Hybrid architectures are a compromise between the deliberative and reactive paradigms, in that they combine the desirable properties of the two approaches. Researchers working upon hybrid agent architectures argue that neither of the two paradigms offer the most effective solution, but rather a combination would (Wooldridge and Jennings, 1995; Fisher, 1995). An analogy for hybrid architectures could be drawn here with the human body in that some actions are triggered by reflex (for example when a hot object is touched) whilst others are planned (for instance, lifting up a cup).

Most hybrid architectures have a tendency to be composed of layers (Ferguson, 1995; Muller *et al.*, 1995; Muller, 1996; Schroeder *et al.*, 1996). Within these layer-based architectures, one set of layers will perform reactive tasks whilst the others will perform deliberative tasks.

Examples of agents based upon hybrid architectures include the Procedural Reasoning System (Georgeff and Lansky, 1987), which has the BDI attitudes of deliberative agents and has been constructed for mobile robots. Another system constructed for robots is Interrap (Muller *et al.*, 1995; Muller, 1996) which simulates robots operating within a loading bay.

Other examples of hybrid architectures include Touring Machines (Ferguson, 1995), where a MAS test-bed is developed and Diagnostic Agent (Schroeder *et al.*, 1996), which also possesses BDI models.

2.1.4 Agent Types and Applications

There are many different types of agents mentioned in the literature, and attempts have been made to categorise them into groups. However, the boundaries between different types of agents are fuzzy as an agent can in general be placed in several different types. As a result of this the type is closely aligned to the application domain within which the agent is being used. The different agent types found in the literature (Wooldridge and Jennings, 1995; Nwana and Ndumu, 1996) include collaborative/co-operative, interface, information/Internet and mobile agents.

Collaborative/Co-operative Agents

The distinction between collaborative and co-operative agents is not necessarily clear as the two terms are often used in an interchangeable manner. However, if separate definitions were to be given, the following could be said. Collaborative agents are agents that interact with other agents in an attempt to accomplish a common set of tasks or achieve a common goal. Co-operative agents, however, interact with each other on rare occasions, to assist each other in completing their own separate tasks, as opposed to achieving an overall goal. Collaborative and co-operative agent research is the most mature research area within the agent field due to the large quantities of work performed by the DAI community. However, despite this it is considered by some that there have been relatively few industrial applications (Nwana and Ndumu, 1996).

In Jennings *et al.* (1993) a collaborative agent system for the control of particle accelerators is proposed. Other applications are in business process management (Alty *et al.*, 1994; Jennings *et al.*, 1996, O'Brien and Wiegand, 1997). Here agent-based systems assist in providing access to documentation relevant to a decision-making process, identifying parties that may be interested in results and informing decision makers about changes that might impinge on the decision-making process. Services within the system are provided by a community of collaborative agents that have had particular tasks assigned to them. Other applications of collaborative agents include a distributed system for the management of patient care (Huang *et al.*, 1995).

Interface Agents

"Interface agent" could be considered a somewhat misleading term as most interface agents do not, as the name suggests, act as an interface between a user and the computer. In reality most successful interface agents are not of this type (Maes, 1994). An interface agent essentially performs the task of a personal assistant which works in collaboration with the user to relieve the need to perform repetitive and time consuming tasks. As such, interface agents can be regarded as assistant agents, which defines their purpose more clearly.

There exist interface agents that perform mainly assisting tasks such as helping the user of a kiosk-based information delivery system (Mamdani and Charlton, 1996). There are agents which assist in the scheduling of meetings (Kozierok and Maes, 1993; Maes, 1994) where an agent learns the user's meeting scheduling patterns and then suggests suitable meeting times. A further progression of this idea is given in Franklin *et al.* (1996) where a clerical agent that creates university seminar schedules is proposed. The agent receives e-mail from seminar organisers, extracts the necessary data, and then checks for any time/room conflicts.

Other tasks performed by interface agents include filtering of newsgroups (Maes, 1994; Sheth, 1994) and e-mail messages (Lashkari *et al.*, 1994; Maes, 1994), where agents search for interesting news, according to criteria specified by the user, and filter important and unimportant e-mail, based upon behaviour patterns, respectively. An application similar to this is performed by the web browsing assistant Letizia proposed in Lieberman (1995, 1997, 1999). This agent system builds a series of heuristics regarding a user's web-browsing habits and searches links within a page to find potentially interesting websites, and then presents a summary of potentially interesting web pages to the user.

Internet/Information Agents

Information agents tend to work in the areas of collecting, manipulating, and processing information from different sources. These sources can take many forms, for example, databases and document repositories. They could be within one system or distributed. The distributed sources could either be distributed throughout one or many organisations, which leads to Internet agents.

Internet agents perform what can be considered to be essentially the same role as information agents. However, they are a specific form of information agents targeted at using the Internet as the source of knowledge or information. This involves them performing similar collection, manipulation and processing functions where the information is distributed amongst many different sites on the Internet.

There are several examples of applications of both information and Internet agents. For example, the RETSINA infrastructure (Sycara, 1999) is a multi-agent system for assisting users in decision-making processes and information-management tasks such as information gathering, filtering and integration.

Several examples exist of agents and their application to business processes. In Papazoglou and van den Heuvel (2000) an agent-based co-operative information system is proposed, and there is also the ADEPT project (Alty *et al.*, 1994; Jennings *et al.*, 1996, O'Brien and Wiegand, 1997), although it is a collaborative agent system, which performs business information-processing tasks on behalf of its users.

Some other examples of information agents include Carnot (Huhns *et al.*, 1992), which is a system that comprises several databases and enables queries to be answered that are not within the scope of any of the databases and the information retrieval agent (Voorhees, 1994), that facilitates the searching of articles held with document repositories.

The distinction between Internet and other agents sometimes is not necessarily clear. For example, some forms of interface agent such as Letizia (Lieberman, 1995) could in their own right be considered as a form of Internet agent, as they collect and process data from a website and provide a concise set of links to websites that the user may wish to follow or be interested in. Internet agents can also theoretically be mobile agents. This means that there is a great deal of overlap between Internet/information and other forms of agent.

An agent that could be termed an Internet-based information agent is Jasper (Davies *et al.*, 1997), which is an agent that attempts to summarise and visualise information stored on the Internet.

Mobile Agents

In Gray *et al.* (1996), the following definition of a mobile agent is given: "Mobile agents are programs that can move through a network under their own control, migrating from host to host and interacting with other agents and resources on each". It has been reported that Maes, in an interview, argued that there is no satisfactory answer to the question "what can you do with mobile agents that you cannot do with stationary agents?". This may be an overstatement but there are many applications that could be achieved effectively using a population of stationary agents.

However, applications of mobile agents exist including those in e-commerce (White, 1994; Andreoli *et al.*, 1997) where agents perform such tasks as finding cheapest prices or negotiating services. Mobile computing is another area where mobile agents have been applied (Gray *et al.*, 1996). Here tasks are performed remotely from mobile computers therefore conserving resources for users of personal digital assistants. Mobile agent-based Internet search engines have also been proposed (Kato *et al.*, 1999).

2.1.5 Machine Learning for Generation of Knowledge Bases

Inductive learning algorithms are algorithms that convert specific data into general rules (Quinlan, 1988; Forsyth, 1989; Hancox *et al.*, 1990). The purpose of inductive learning can be considered to be two-fold, firstly to perform a synthesis of new knowledge, which is independent of the form of the original input data (Kodratoff, 1988) and secondly to ease the knowledge acquisition bottleneck that occurs when knowledge-based systems are developed.

An inductive learning algorithm will typically either produce a rule set of IF…THEN rules or a decision tree (Figures 2.2, 2.3). However, the decision tree generated by the algorithm can be easily converted into a set of production rules (Al-Attar, 1991; Quinlan, 1987). This is because every branch of a decision tree can be considered as an IF…THEN rule. The rule set derived from the decision tree illustrated in Figure 2.2 is shown in Figure 2.3. It should be noted that it may not necessarily be possible to convert a rule set into a decision tree.

Inductive learning algorithms can take two different forms, non-incremental, or batch, and incremental. Non-incremental algorithms use the entire set of training examples to generate a rule set. If a new example is added, the entire historical set of examples plus the new example must be used to generate the new rule set. With incremental learning algorithms, only the new example need be presented to the algorithm. There are several advantages in the use of incremental learning algorithms including being able to update stored knowledge rapidly and to perform on-line learning (Schlimmer and Fisher, 1986). However, incremental algorithms only makeup a minority of the inductive learning algorithms in use (Pham and Dimov, 1997b).

There exist several well-known families of inductive learning algorithms, which contain both the incremental and non-incremental types. Some of these families of algorithms are examined and described below.

AQ family

The AQ family includes AQ11 (Michalski, 1983), CN2 (Clark and Niblett, 1989; Clark and Boswell, 1991), and the incremental AQ15 (Michalski *et al.*, 1986) learning algorithms.

Members of the AQ family generate sets of decision rules from a series of instances or examples presented to the algorithm. The AQ family is based upon the "extension-against" approach. This means a hypothesis rule space is generated based upon an example set that is refined to find more general rules.

The AQ algorithm performs a heuristic search through a space of logical expressions or descriptions that cover only examples of the one selected type of class (positive examples). Then, according to a set of criteria, the final rule is chosen. AQ15 (Michalski *et al.*, 1986) when being employed for incremental learning performs this operation with what is termed a "perfect memory" in that both rules and examples used to create rules are recorded. This also enables the user to specify rules *a priori* then the algorithm can build upon these rules if required.

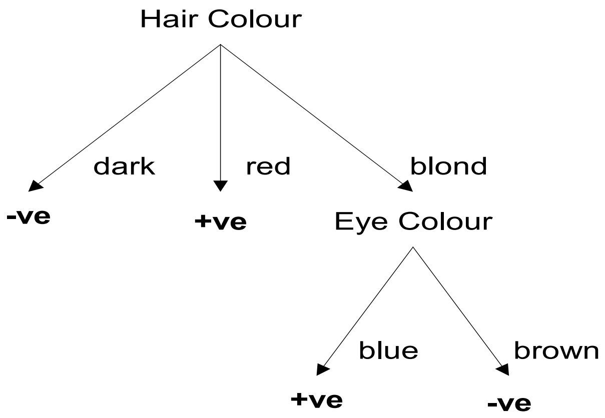


Fig. 2.2 Example decision tree

IF Hair Colour = dark THEN – ve
IF Hair Colour = red THEN + ve
IF Hair Colour = blond AND Eye Colour = blue THEN + ve
IF Hair Colour = blond AND Eye Colour = brown THEN – ve

Fig. 2.3 Rule set for example decision tree

The rule generation process of the AQ family can basically be summarised as follows (Aksoy, 1993):

1. Select an unclassified example. The class of this instance is considered to be positive whilst the others are negative.
2. Apply the "extend against" approach to produce a complete set of descriptions for the current instance.
3. From the set of descriptions produced in the 2nd step, a description is selected based upon its consistency (related to a description and negative classes) and completeness (related to a description and positive classes).
4. If the description is complete, i.e. it covers all positive classes, then go to the 6th step.

5. If not, then reduce the size of the positive set to include only those instances which are not covered by the description. Go to the 1st step.
6. Convert all descriptions into a rule that can cover all examples in the positive class. If no more unclassified examples exist then stop. Otherwise go to the 1st step again.

ID3 Family

The ID3 family of learning algorithms is based upon the ID3 learning algorithm (Quinlan, 1986, 1987, 1988), which itself is based upon the concept learning system (Hunt *et al.*, 1966). The family includes the ID4 (Schlimmer and Fisher, 1986) and ID5 (Utgoff, 1988) incremental learning algorithms. The ID3 family also includes the C4.5 and C5.0 learning algorithms.

The ID3 family of algorithms generate multi-branched decision trees from the set of examples that they are presented with. ID3 uses an information theory approach to decision tree generation in an attempt to minimise the number of tests required to classify an object (Quinlan, 1986). The ID3 family follow what is known as the top-down approach, which refers to the breaking down of a set of examples into several subsets until there remains a unique class in each of these subsets. The rule generation method of ID3 can be expressed as a basic number of steps that describe the process involved.

1. Select an attribute to divide the set of examples into subsets.
2. If there are subsets containing examples in different classes, return to step 1.
3. Stop when, within each subset, there are examples in one class only.

A modified version of ID3, ID3-IV (Quinlan, 1986), named in Cheng *et al.* (1988), which includes a function for calculating information gain was produced, as ID3 has a tendency to choose condition attributes that have more values, thus causing generic ones to be missed. Another version for solving the irrelevant condition problem (gives rise to rules testing unnecessary attributes), called GID3 or generalised ID3 (Cheng *et al.*, 1988), also uses the idea of the information gain ratio.

The ID4 (Schlimmer and Fisher, 1986) approach of generating decision trees differs from the approach adopted by ID3. As opposed to using a set of examples, ID4 generates and updates the tree based upon each example within the data set. At each node in an ID4 decision tree there is located a table, within which are contained values of non-test attributes and the numbers of positive and negative examples. When the information measure is calculated for a non-test attribute and it transpires that it is lower than the value for the current test attribute, the sub-trees below this node are deleted and the node itself is modified.

ID5 (Utgoff, 1988) differs not only from ID3, but also from ID4. ID5 builds on the concept introduced with ID4 of keeping counts of positive and negative examples of each attribute that could be used to create a test attribute (Utgoff, 1988). The principal difference between ID4 and ID5 relates to the methods used for the replacement of test attributes. Instead of the method of discarding the sub-trees below the test attribute, the decision tree is reorganised by adding or changing the positions of current nodes.

The C4.5 algorithm has essentially the same structure as the ID3-IV algorithm and is considered to be the “industrial” version of the algorithm. However, there have been several modifications. For example, C4.5 has decision tree pruning features and, once a tree has been pruned, it is converted into rule sets. Also, C4.5 is able to deal with continuous numbers, noise, and missing values. C5.0 is a modified version of C4.5, with new features added including variable misclassification costs and additional data types.

RULES Family

The RULES family of algorithms consists of RULES-1, RULES-2, RULES-3 (Pham and Aksoy, 1993, 1995), RULES-3+ (Pham and Dimov, 1997a) and RULES-4 (Pham and Dimov, 1997b), which is the incremental version of the algorithm. The RULES family was developed in the author's laboratory.

The rule-forming procedures of the different variants of the RULES family all work in slightly different ways, except for RULES-3+ and RULES-4 where the processes are essentially the same. Basically the algorithm takes an example from a set of unclassified instances, then attempts to form a set of objects (proto-rules) for it using different combinations of conditions based upon the attributes and the values in the example. Objects which do not belong to the class of the example are removed. From the remaining objects the best rule is selected. Then all examples that are classified by this rule are removed from the list of unclassified examples.

The following gives an overview of the rule generation process of the RULES-3 learning algorithm:

1. Define the ranges for continuous attributes (a user action).
2. Specify the minimum number of conditions for each rule (a user action).
3. Number_Conditions = minimum number of conditions −1.
4. Take an unclassified example.
5. If Number_Conditions is less than Number_Attributes, then increment Number_Conditions by one.
6. Get all values and attribute labels for example.
7. Form objects that consist of Number_Conditions, conditions, using values and labels in the example.
8. If one or more objects belong to a unique class then form rules with them, ELSE go to the 5th step.
9. Select the rule that classifies the most examples.
10. Remove these examples from the list of unclassified examples.
11. If there are no more unclassified examples then stop. Otherwise go to the 3rd step.

The rule-forming procedure of RULES-3+ and RULES-4 differs from that for RULES-3. This derives mainly from the fact that the information content, generality and accuracy of rules are used in the algorithms (Pham and Dimov, 1997b).

The incremental learning procedure in RULES-4 uses two forms of memory, the long-term memory (LTM) and short-term memory (STM), which contain the generated rules and a selection of the examples used to form the rules. Firstly when an example is chosen a check is performed as to whether the rules in the LTM classify the example. At this stage the accuracy and information content measures are updated and the LTM is pruned if the accuracy falls below a given threshold. Then, depending on whether there are free spaces in the STM and if the example is not covered by the LTM, the example is added to the STM. If an example in the STM is not covered by the LTM then the rule-forming procedure is applied, and the new rule is added to the LTM, until there are no examples uncovered by the LTM.

2.2 Product Fault Knowledge Acquisition and Management

2.2.1 Automating Knowledge Base Management

A suitable means for the automation of the knowledge base management process is the combined use of an agent-based system together with machine learning techniques for knowledge base generation. Agent-based systems as shown in the previous section are ideally suited to information-processing tasks and have been employed in many different information-processing applications (Alty *et al.*, 1994; Etzoni and Weld, 1994; Maes, 1994; Sheth, 1994; Jennings *et al.*, 1996; Davies *et al.*, 1997; O'Brien and Wiegand, 1997; Papazoglou and van den Heuvel, 2000; Leiberman, 1999; Pham *et al.*, 2001). Agent systems are also ideally suited to operating within distributed environments (Huang *et al.*, 1995; Jennings and Wooldridge, 1998). Machine learning techniques have a long history of being applied to the generation of rule sets for use in knowledge-based systems as a means of reducing the bottlenecks which occur when creating such systems (Pham and Dimov, 1997b; Quinlan, 1988).

The agent-based system proposed within this chapter performs both user assistance and information gathering and processing and could therefore be perceived as being an interface agent.

The proposed agent-based system could be developed to adopt either an approach that uses the ISLE rule extraction algorithm, or to utilise an approach involving a combination IFCLEAR and RULES-4 (or ISLE without its unclassified example abilities). As the intention is also to provide a generic framework to allow any learning algorithm to be employed (such as ID3 or C5.0), it was decided to implement an agent-based system that encompassed both approaches thus increasing the potential applicability of the proposed architecture.

2.2.2 Analysis of Knowledge Base Management Process

To be able to develop an agent-based system for the management of expert system knowledge bases, the manual processes involved in the knowledge gathering and processing procedures should be known. To do this, tasks and steps that are carried out in real life are analysed. From this analysis, the actual tasks that need to be performed by the different agents within the system can be determined and specified.

The decision-making and physical processes involved should therefore be understood before the structure and functionality of the agents can be specified. As such, the workflow processes for gathering and processing fault data are generated from an analysis of the processes involved at these two stages. It is essential to separate these processes even though the latter is dependent upon the former. This is because they take place at different locations and involve separate organisations.

Therefore the following two sections analyse and define business processes for the collection and processing of fault data respectively, taking into consideration the people, actions and decision-making processes involved at the various stages.

Collection of Fault Data

The process for a service technician compiling the fault report illustrated in Figure 2.4 can be considered relatively straightforward. However, there is the problem of report forms not being completed due to procrastination. An equally serious problem arises during the subsequent processing of the forms if they have not been completed correctly.

The first task a service technician must perform is to determine what type of fault report he has to provide: a new account of an existing class of fault, a description of a new type of fault or a report on a fault relating to attributes that are not included within the present knowledge base.

If the report contains a new fault or provides a new account of an existing fault then the service technician will need to specify the fault and the conditions that cause it. If a new attribute has to be defined because the current knowledge base does not contain it, details of the new attribute along with the relevant information would normally need to be specified.

Once the process of actually gathering fault information has been completed the report has to be forwarded to the manufacturer. This is another situation where problems can arise, namely that the report documents can be mislaid (either at the maintenance or manufacturing site) or perhaps not forwarded to the manufacturer. Therefore, potentially valuable information regarding product faults might be lost through elementary mistakes.

Generation and Management of Knowledge Bases

The procedure involved in the processing and management of fault data has been simplified with the development of a parser to convert a spreadsheet table containing both training and attribute data into the data and format files typically required by learning algorithms. This removes the need to perform the conversion manually. The files contain the data used to create the final rule set and information concerning what this data describes. The action of this parser is illustrated in Figure 2.5.

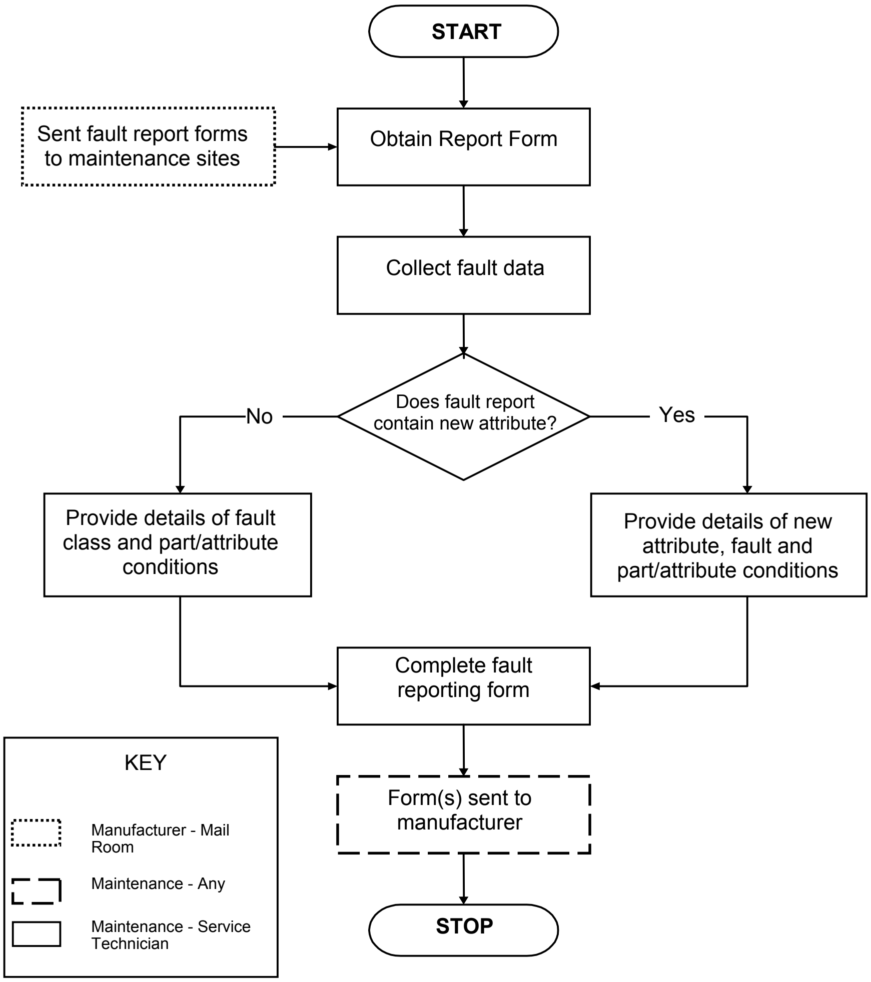

START
Sent fault report forms to maintenance sites
Obtain Report Form
Collect fault data
Does fault report contain new attribute?
No
Yes
Provide details of fault class and part/attribute conditions
Provide details of new attribute, fault and part/attribute conditions
Complete fault reporting form
Form(s) sent to manufacturer
STOP
KEY
Manufacturer - Mail Room
Maintenance - Any
Maintenance - Service Technician

Fig. 2.4 Tasks involved in providing fault data

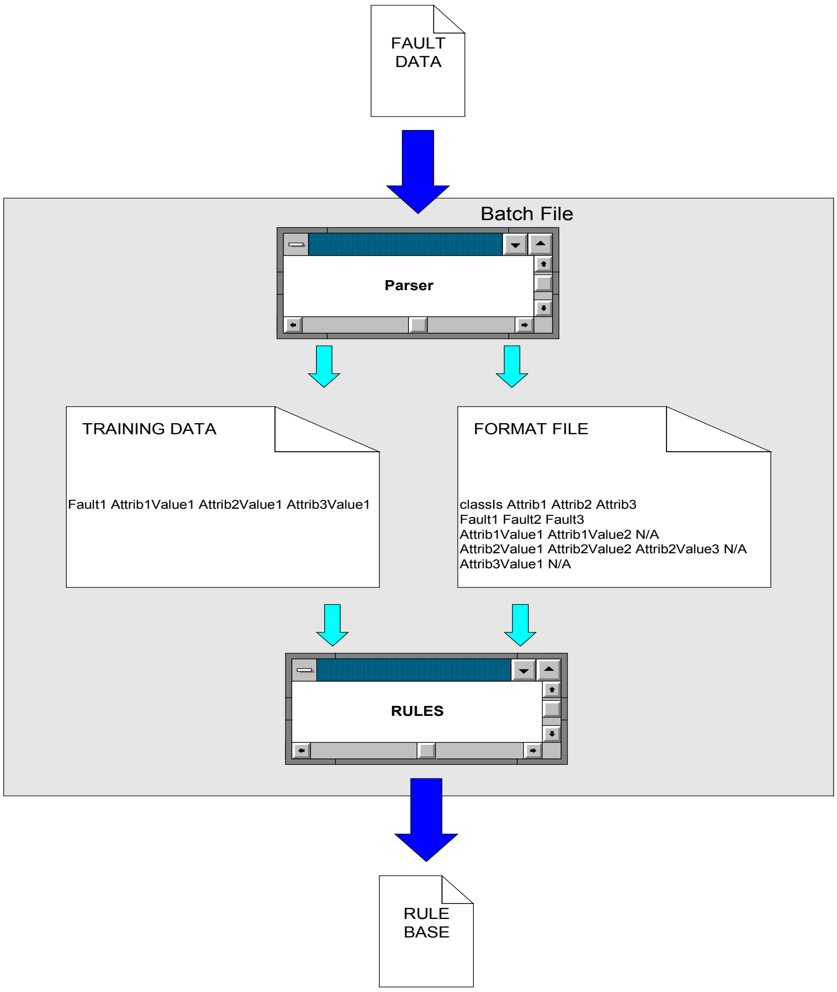

FAULT DATA
Batch File
Parser
TRAINING DATA
Fault1 Attrib1Value1 Attrib2Value1 Attrib3Value1
FORMAT FILE
classIs Attrib1 Attrib2 Attrib3
Fault1 Fault2 Fault3
Attrib1Value1 Attrib1Value2 N/A
Attrib2Value1 Attrib2Value2 Attrib2Value3 N/A
Attrib3Value1 N/A
RULES
RULE BASE

Fig. 2.5 Generation of rules from fault data file

Once a manufacturer has received a fault report, a more complex procedure and business process must be followed. The workflow for these tasks is illustrated in Figure 2.6. First, the fault report would be received in the mailroom (or its equivalent) within the organisation. The report then ideally would be passed to an administrative assistant who would ascertain whether it is legible and therefore useable. If not, depending upon its procedures, the company may wish to request that the form be resubmitted.

An acceptable fault report is then passed on to a suitably qualified and experienced engineer for processing. The data must then be checked to discover whether all attributes have been instantiated. If this is not the case, then a resubmission request is made as, without these items of data, the new scenario described in the fault report may be of little or no value. If the data is complete then a check is performed as to whether this fault is already covered by the knowledge base or instance history. If so, there is no need for further work as this fault is already known about. Otherwise the procedure continues.

The fault report is then checked to see if the fault class has been specified. If not, then it must be found. This task would typically be done by the engineer and might utilise several possible factors including his expert knowledge of the product and faults that have occurred with similar products or be based on attempting to replicate the fault. If the fault class was specified or discovered then the data pertaining to the new fault report is entered into a spreadsheet. At this stage, consistency checks have to be performed to ensure that the attributes and their values are correctly spelt and that the spreadsheet table is correctly formatted. This is particularly important as otherwise problems may arise, for example, an inaccurate rule set is generated or errors occur during the running of the machine learning algorithm itself. The data is then saved in a format appropriate for the parser. As illustrated in Figure 2.5 the parser is executed, which takes the fault data and generates the training and format data required by the machine learning algorithm. The latter in turn generates the knowledge base. The old knowledge base must be deleted and replaced with the newly created knowledge base.

Knowledge Base Management Issues

The above discussion has highlighted the work involved in the various tasks relating to knowledge base management. From this analysis, it can be determined that there are several problems inherent in the manual process that can potentially be overcome or mitigated by automation. The problems relate both to inaccuracies in the data and inefficiencies of certain processes resulting in bottlenecks within the overall knowledge management procedure. These problems, which have been already mentioned in different places, are summarised below.

- Failure to fill-in fault report forms: this might be caused by forgetfulness, procrastination, or an unwillingness to carry out the task.
- Non-receipt of report forms by manufacturer: this might arise from the forms not being sent or having gone astray during posting.
- Illegible fault report forms: this might be caused by untidy handwriting or oil, grease and dirt marks; consequently the data either is unusable or requires a large amount of time to interpret.
- Incomplete data: this might be due to carelessness during form completion.

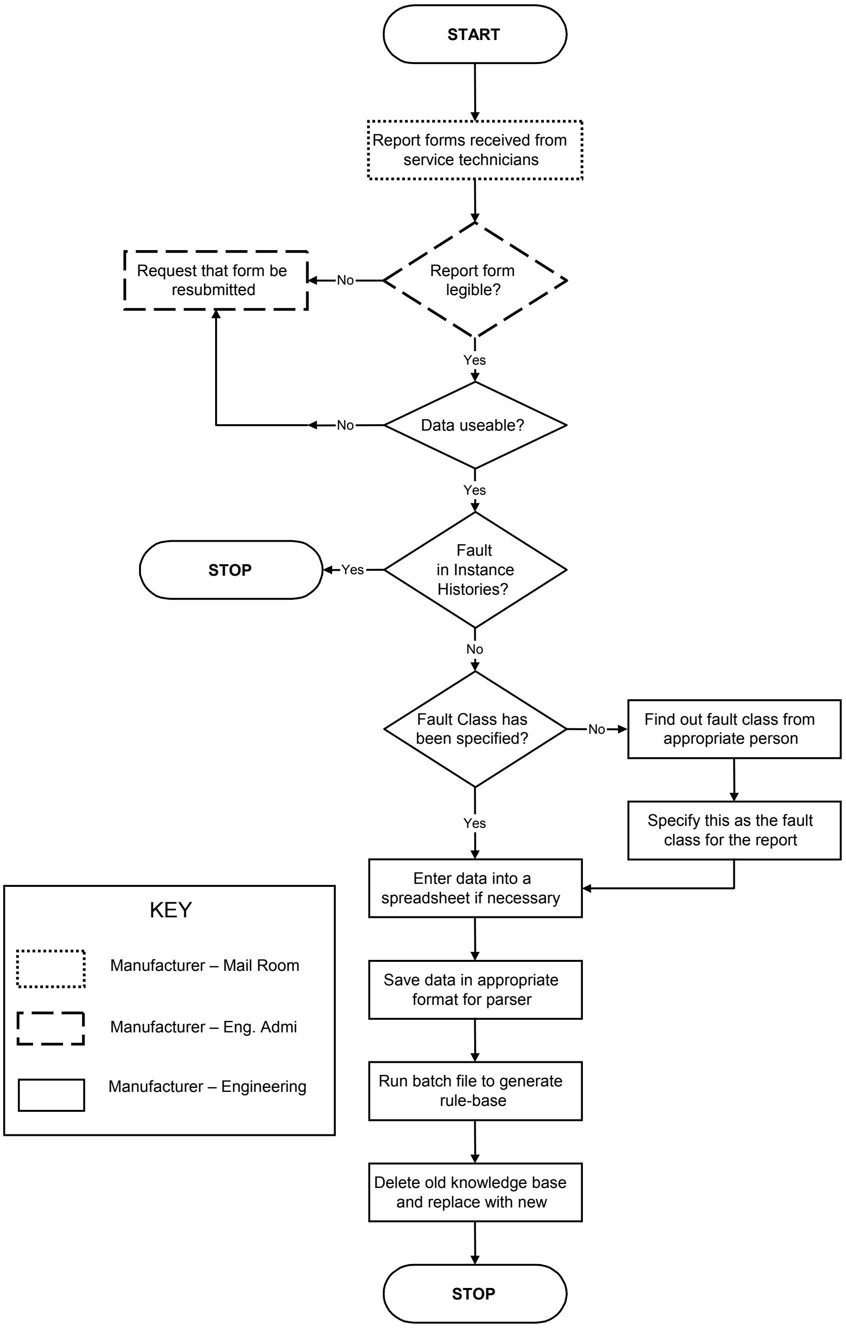

START
Report forms received from service technicians
Report form legible?
No
Request that form be resubmitted
Yes
Data useable?
No
Yes
Fault in Instance Histories?
Yes
STOP
No
Fault Class has been specified?
No
Find out fault class from appropriate person
Specify this as the fault class for the report
Yes
Enter data into a spreadsheet if necessary
Save data in appropriate format for parser
Run batch file to generate rule-base
Delete old knowledge base and replace with new
STOP
KEY
Manufacturer – Mail Room
Manufacturer – Eng. Admi
Manufacturer – Engineering

Fig. 2.6 Process – manufacturer's side

- Lengthy movement of reports: the fault reports would typically pass through the hands of several people before reaching the relevant engineer, thereby wasting time and also increasing the risk of documents being delayed or misplaced.
- No guarantee as to when a report will be processed: the engineer responsible may be unavailable.
- Long and tedious data entry: entering data from a received report into a spreadsheet requires translation of data into an electronic form and is potentially very time consuming.
- Need to perform consistency checks and corrections: the data entered has to be checked to ensure that attributes and their values are consistent throughout the spreadsheet. This takes time particularly when data sets are large.
- Need to ensure the data is formatted correctly: this is similar to above but involves checking and arranging that attributes and their values are in the correct places within a document.

2.3 Agent System for Knowledge Acquisition and Management

The proposed agent-based system for collecting and processing new fault data consists of two agents, the server agent (SA) and user agent (UA), as illustrated in Figure 2.7. These agents perform two separate but interdependent roles, much as in the business processes discussed in the previous section.

From the illustration in Figure 2.7, it can be envisaged that UA operates within the end-user's computer and interacts with him. A “virtual form-filling” session collects and sends new fault reports to SA via a BB. SA then processes this new fault data and incorporates it into the knowledge base. The system is structured such that it allows SAs that manage the knowledge bases for different product variants to communicate with one another if necessary, in order to facilitate the retrieval of fault classes.

In the implementation of this proposed application, there is a requirement for communication between agents that operate at geographically distributed locations. This communication is envisaged to take place via local-area or wide-area networks, such as an Intranet and the Internet respectively.

To meet these requirements, this system is implemented using Sun Microsystems Java 1.1 programming language. Java was chosen as it is very network/Internet oriented and is platform independent, which is very important since the intelligent product manual (IPM) is intended as a platform-independent solution. Of particular use are the remote method invocation (RMI) features of Java, which facilitate the construction of agent-based systems. RMI allows Java-based programs to communicate with one another across a local-area network or a wide-area network. Java provides a useful means to develop small applications (applets), which are capable of being executed from within web pages viewed in a web browser such as Internet Explorer or Netscape. Since the IPM uses Internet-based delivery (Pham *et al.*, 1998), this ability is of particular importance as it allows UA to be embedded within an HTML page in the user's computer.

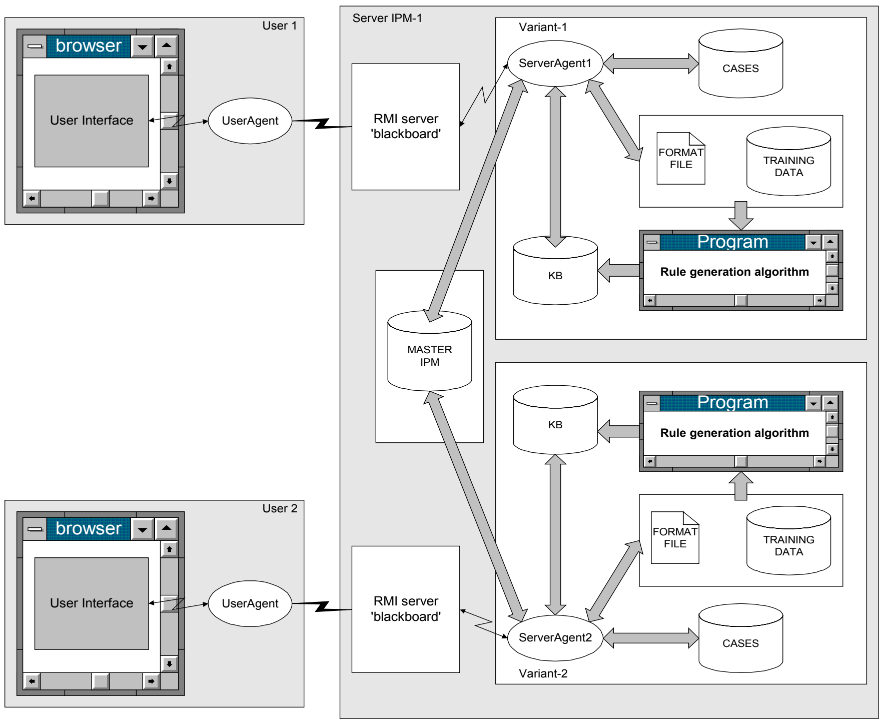


Fig. 2.7 Agent system

The system as a whole is not purely agent based but can be considered to be a hybrid of an agent-based system and a BB system in a manner similar to Lander *et al.* (1990), Hayes-Roth *et al.* (1994) and Holt and Rodd (1994). This is because there are certain BB features and concepts that are particularly appropriate to this system. Also RMI makes the system lean towards being BB based. The feature of BB systems that is of particular relevance is the globally accessible data structure known as a BB as mentioned in Englemore and Morgan (1988) and Craig (1995).

The utilisation of concepts from BB systems manifests itself in the BB server that is used as a channel or medium for all communications. The BB server receives and queues all messages sent to a particular SA in a first-in-first-Out (FIFO) manner. The FIFO strategy is the most logical in this application as it helps ensure that no request will wait a disproportionately long period before it is processed. The use of the BB server also removes the need for SA to perform any message management and thus frees SA to concentrate upon performing information-processing-related tasks. Once SA has finished a task, it can request that the next message in the queue be forwarded. If there are no messages then SA goes into a wait state until it is given a new task to perform.

An additional benefit of this method is that it allows SA to be located on a machine remote from the BB server. For example, the BB server can be placed on an Intranet/Internet server and SA can be running on another machine that communicates with the BB server via the local area network or the Internet. Indeed, the IPM itself could be placed on a third machine if necessary. This allows a distributed system architecture to be used if deemed necessary, thus spreading processing burdens across several machines. Theoretically, if such a distributed approach is used the degree of fault tolerance and robustness with respect to problems that might occur with SA can be improved as the BB server can keep operating whilst SA is unavailable.

2.3.1 User Agent

The purpose of UA is to facilitate the acquisition of new fault data from service technicians. This is achieved by UA providing a user interface for the entry of new fault examples where the user partakes in a "virtual form-filling" session. The functionality of UA derives from the business process for the collection of fault reports analysed previously.

Task Analysis

The majority of the tasks involved in UA collecting new fault data from the user (Figure 2.8) are very similar to those in the process illustrated in (Figure 2.4). There are, however, additional actions taken by UA that are necessary to ensure the complete collection of data from the user. In particular, UA determines when all attributes have been instantiated.

The user will activate UA, which initially will obtain attribute and legal value information from the BB server. Once this has been done UA will need to know whether the end user is specifying a new attribute and fault as opposed to just a new fault scenario. If a new attribute and fault scenario are being specified then additional information such as part (or attribute) name, current condition and other known part condition values must be entered.

The fault class and attribute values are then collected from the end-user (i.e. the service technician). Because it is necessary for all attributes to have a value, the interface needs to be designed in such a manner that all attributes used are assigned a value (even it is N/A for not applicable/available). UA gathers the information until all attributes have been instantiated. Once this has been done, the information regarding the fault is passed to the BB server and then the connection to the BB server closed.

Architecture

As a result of the highly procedural nature of the process shown in Figure 2.8), the agent is essentially rule-driven. For example, the process can be specified in terms of such statements as “IF new fault scenario THEN obtain only attribute values” and “IF attribute instantiated THEN proceed to next attribute”. However, instead of adopting an explicit rule-based representation where UA uses a rule base to express the functions that it will perform, the rules are actually embodied in the code for UA. As such the operation of UA is essentially hardwired into the code of UA.

Whilst this is perhaps not the most elegant or flexible solution, there are several factors which necessitate it. This is very much due to a need to the have a trade-off between the complexity of the operation and implementation and the size of the UA Java applet. The latter issue is particularly important as the potential range of network connection speeds has to be taken into consideration.

It is highly feasible that the link between the site where a service technician is operating and the manufacturer may be via an Internet service provider and hence may employ the standard telecommunications network. Therefore, modems with operating speeds no higher that 56 kbps (kilobits per second) will probably be used. This means that ideally the applet should be engineered to be small, resulting in a minimal amount of transmission taking place to reduce any delays caused by the speed of the Internet link. It is therefore highly advantageous that the agent does not have to load any more information than is absolutely necessary. The size of the applet is sufficiently lightweight at only 7 kB, or 57,344 bits, therefore achieving download times that could be considered to be acceptable.

The nature of UA and its implementation mean that its architecture does not necessarily prescribe to one of the many different agent architecture types given in earlier in the chapter. However, it could be considered to be of a reactive nature as no planning can be said to occur and thus it could perhaps reasonably be described as a procedural agent. Despite the fact that UA performs what can be considered very elementary tasks, it does exhibit some characteristics of an intelligent or autonomous agent.

Operation

UA is downloaded as a Java applet embedded within an HTML web page residing on a web server. Once this applet has been downloaded and initialised, it opens a connection and registers itself with the BB server. Upon the completion of the registration process, UA automatically receives a custom Java object of type “Legal-Values” that contains a list of the current attributes and their legal values. The user interface of UA, shown in (Figures 2.9 and 2.10), is then displayed within the Internet browser.

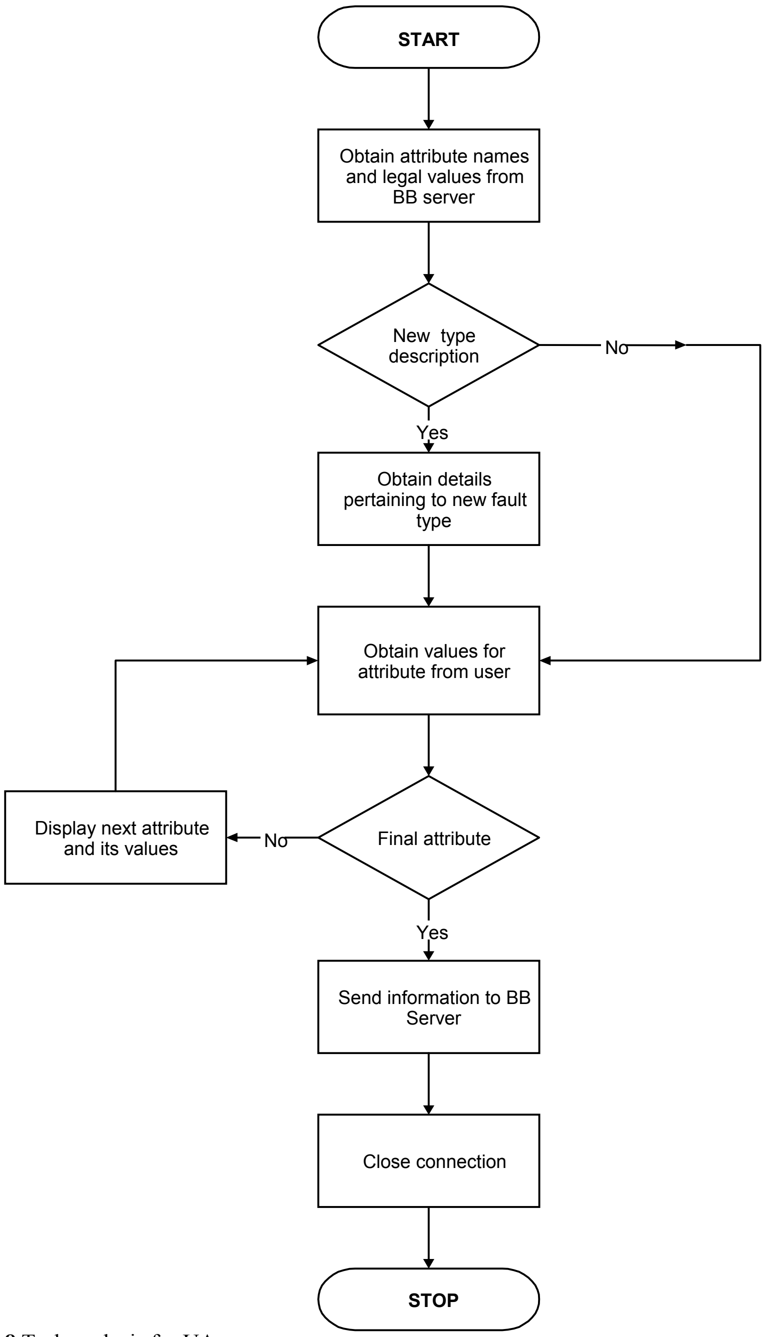
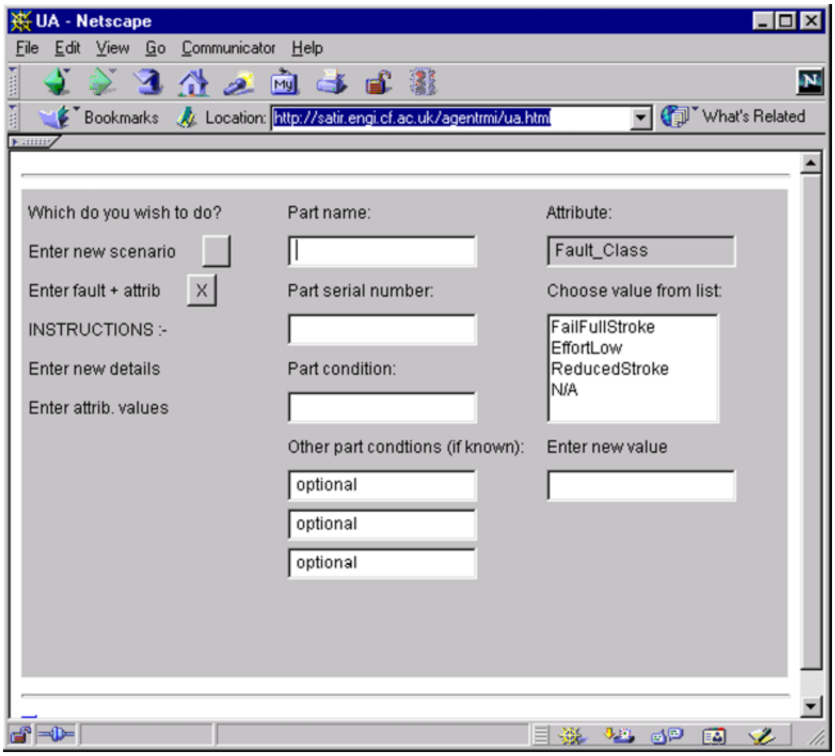

START
Obtain attribute names and legal values from BB server
New type description
No
Yes
Obtain details pertaining to new fault type
Obtain values for attribute from user
Final attribute
No
Display next attribute and its values
Yes
Send information to BB Server
Close connection
STOP

Fig. 2.8 Task analysis for UA

The user interface, consists of essentially three panels:

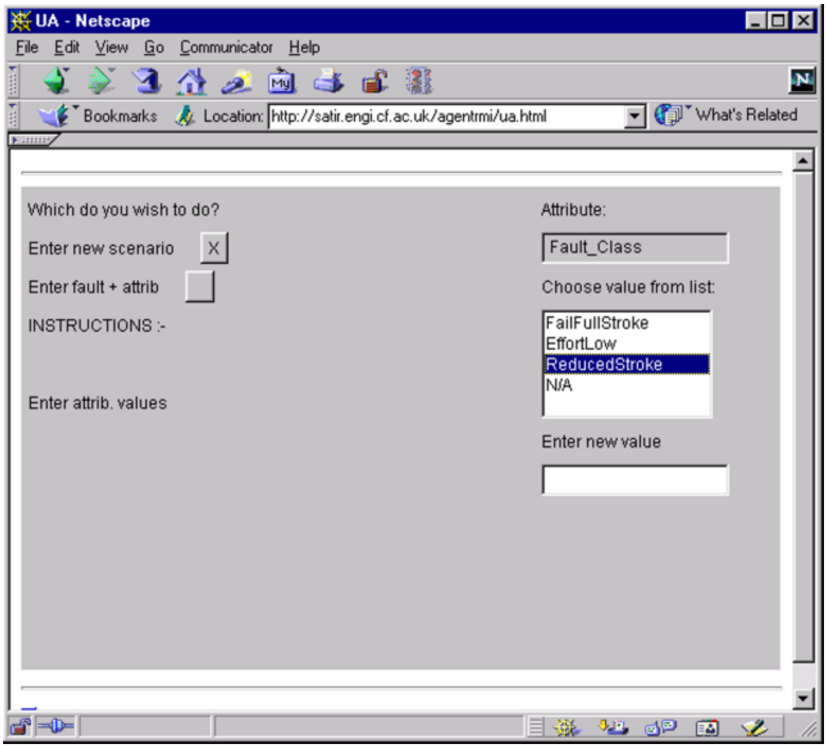
- *Options (left-hand side)*: this panel allows users to select whether they wish to enter a description of a new fault that contains new attributes (Figure 2.9) or to input a new fault scenario, which is a new description of an existing fault type (Figure 2.10). This is achieved by activating the appropriate button on the panel, either the Fault + Attrib or New Scenario button respectively.
- *New Attribute Details (middle)*: this panel allows the user to define a new attribute and its values if required. It would be employed when a fault involves a part or attribute that is not included in the current knowledge base. Otherwise this panel is not visible as can be noted in (Figure 2.10).
- *Fault Details (right-hand side)*: this panel provides a standardised means for entering the details of a new fault. Initially, UA asks for the fault class. It then proceeds to request all of the part or attribute conditions. The end-user can either choose these from the list shown or input a new value in the text-entry area below the list of values.



http://satir.engi.cf.ac.uk/agentrmi/ua.html

Which do you wish to do?	Part name:	Attribute:
Enter new scenario		Fault_Class
Enter fault + attrib X	Part serial number:	Choose value from list:
INSTRUCTIONS :-		FailFullStroke EffortLow ReducedStroke N/A
Enter new details	Part condition:	
Enter attrib. values		
	Other part condtions (if known):	Enter new value
	optional	
	optional	
	optional	

Fig. 2.9 Initial screen/entering fault containing new attribute



http://satir.engi.cf.ac.uk/agentrmi/ua.html

Which do you wish to do?

Enter new scenario [✓ X]

Enter fault + attrib

INSTRUCTIONS :-

Enter attrib. values

Attribute:

Fault_Class

Choose value from list:

FailFullStroke
EffortLow
[✓ ReducedStroke]
N/A

Enter new value

Fig. 2.10 Entering new scenario screen

As can be seen from both (Figures 2.9 and 2.10), the user interface of UA does not contain the usual OK button. This omission may make the interface less intuitive than it could be, but its purpose is to help ensure that values are specified for all attributes. With such an arrangement, the user would have to either double-click on an item in the list or enter a value manually and then press the “return” button before the value would be accepted. This forces him to instantiate all of the attributes used.

From this description of the user interface and its operation, it is apparent that UA has two modes of data entry available:

- *Description of new fault and attribute*: this involves using the middle panel to specify the details of a new attribute (system, sub-system, part, etc.) that is to be included within the knowledge base. The right-hand panel is employed to enter or select the fault class and conditions of attributes within the system. This process of entry is illustrated in Figure 2.11. In the first screen, the end-user provides information relating to the new attribute (part) that is to be entered into the knowledge base by specifying its name, part number and conditions. Next, the fault class is entered. This can pertain to either a new fault or an existing fault. The end-user then proceeds to specify the values of the other attributes. These can be either selected or manually entered. Once the final attribute has been instantiated, the left-hand panel is not needed and thus no longer appears on the user interface.

- *Entering a new fault scenario*: this procedure is similar to that adopted for new faults and attributes. The primary exception is that no new attributes (parts) are specified and only values for the fault and attribute are selected or entered. This process is illustrated in Figure 2.12. Initially, the fault class is selected or entered, followed by the attribute values that describe this fault. As before the attribute values can either be chosen from a list or manually specified. At the end of the fault-reporting session, the left-hand panel of the user interface is removed.

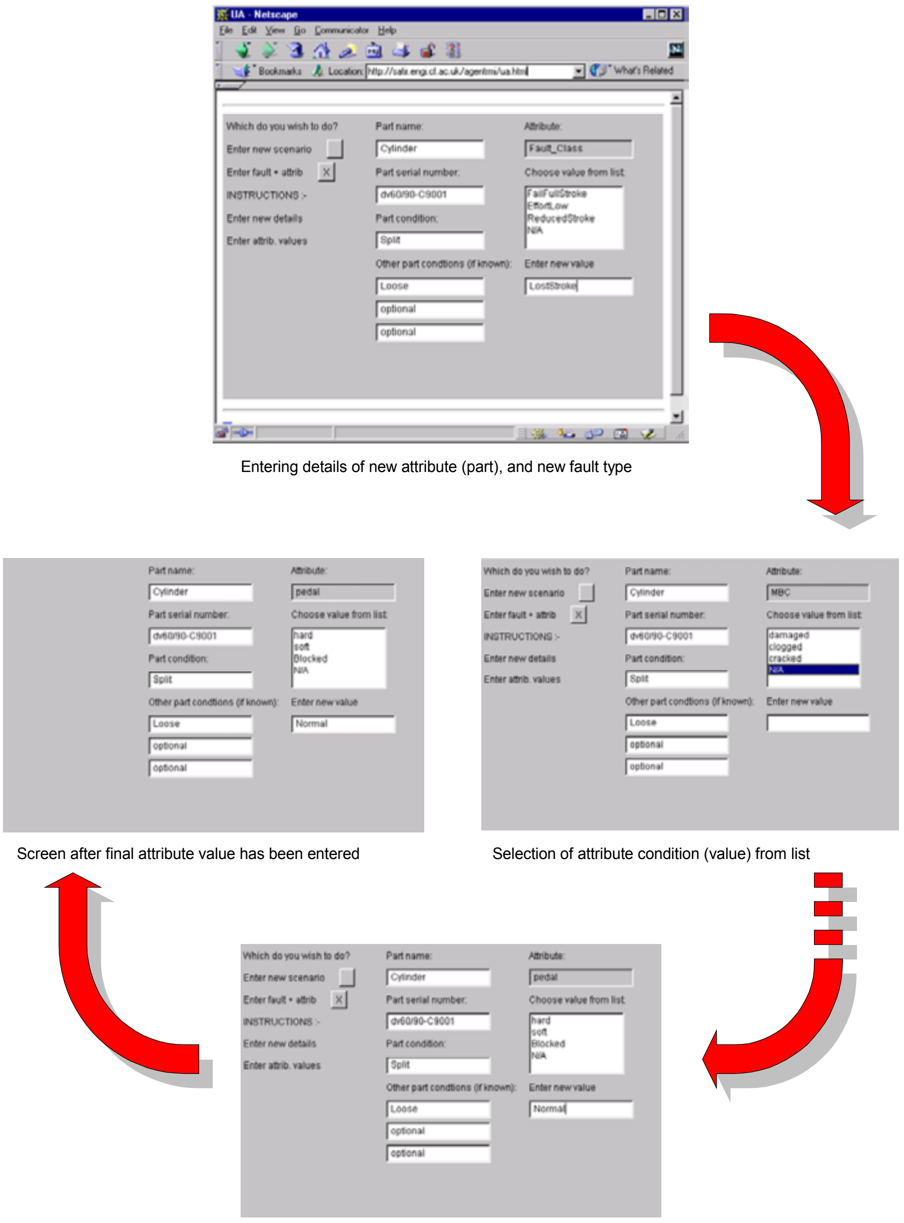
After the service technician has completed the "virtual form-filling" process UA returns a custom Java object of type "Fault" to the BB server. Here, the fault report awaits collection by SA so that it can be processed. The network links between UA and the BB server are then disconnected.

2.3.2 Server Agent

The purpose of SA is to manage the generation and modification of the knowledge bases within an IPM. The new fault details are passed to SA from UA. SA then processes the fault data and, if appropriate, generates a new knowledge base using an inductive learning algorithm.

Task Analysis

The task analysis produced for SA is understandably more involved than that for UA, due to the greater complexity of the relevant business process (Figure 2.6). However, as mentioned previously, for this task analysis there are potentially two differing approaches that can be followed. A conventional learning algorithm such as RULES-4 (or ISLE but without its ability to handle unlabelled instances) combined with the IFCLEAR algorithm detailed in Pham and Soroka (2006) could be used. Alternatively, ISLE or RULES-IS (Pham and Soroka, 2005ab, 2006) with all its features could be employed. Despite this, the basic processes involved will be nearly identical for both approaches. Therefore, only one task analysis, that for the pure ISLE-based approach, is presented with the non-common areas highlighted.


UA - Netscape
File Edit View Go Communicator Help
Bookmarks Location: http://sals.engi.cf.ac.uk/agentms/ua.html
What's Related

Which do you wish to do? | Part name: Cylinder | Attribute: Fault_Class
Enter new scenario | Part serial number: dv60/90-C9001 | Choose value from list: FailFullStroke, EffortLow, ReducedStroke, N/A
Enter fault + attrib [✓ X]
INSTRUCTIONS :-
Enter new details | Part condition: Split
Enter attrib. values | Other part condtions (if known): Loose, optional, optional | Enter new value: LostStroke

Entering details of new attribute (part), and new fault type

Which do you wish to do? | Part name: Cylinder | Attribute: MBC
Enter new scenario | Part serial number: dv60/90-C9001 | Choose value from list: damaged, clogged, cracked, [✓ N/A]
Enter fault + attrib [✓ X]
INSTRUCTIONS :-
Enter new details | Part condition: Split
Enter attrib. values | Other part condtions (if known): Loose, optional, optional | Enter new value:

Selection of attribute condition (value) from list

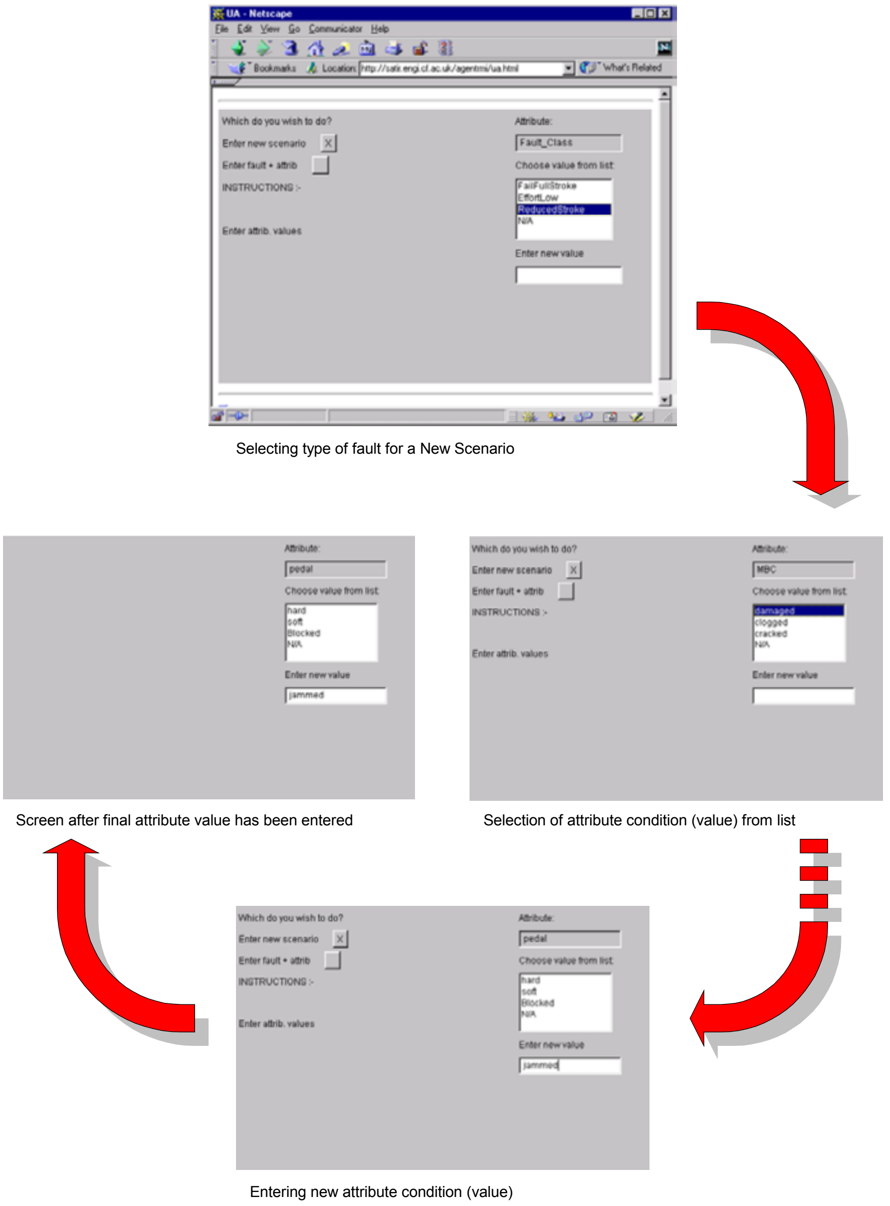
Which do you wish to do? | Part name: Cylinder | Attribute: pedal
Enter new scenario | Part serial number: dv60/90-C9001 | Choose value from list: hard, soft, Blocked, N/A
Enter fault + attrib [✓ X]
INSTRUCTIONS :-
Enter new details | Part condition: Split
Enter attrib. values | Other part condtions (if known): Loose, optional, optional | Enter new value: Normal

Entering new attribute condition (value)

Part name: Cylinder | Attribute: pedal
Part serial number: dv60/90-C9001 | Choose value from list: hard, soft, Blocked, N/A
Part condition: Split
Other part condtions (if known): Loose, optional, optional | Enter new value: Normal

Screen after final attribute value has been entered

Fig. 2.11 Entering a fault description that contains a new attribute


UA - Netscape
File Edit View Go Communicator Help
Bookmarks Location: http://safir.engi.cf.ac.uk/agentmi/ua.html
What's Related

Which do you wish to do?
Enter new scenario X
Enter fault + attrib
INSTRUCTIONS :-
Enter attrib. values
Attribute: Fault_Class
Choose value from list: FailFullStroke, EffortLow, ReducedStroke, N/A
Enter new value

Selecting type of fault for a New Scenario

Which do you wish to do?
Enter new scenario X
Enter fault + attrib
INSTRUCTIONS :-
Enter attrib. values
Attribute: MBC
Choose value from list: damaged, clogged, cracked, N/A
Enter new value

Selection of attribute condition (value) from list

Which do you wish to do?
Enter new scenario X
Enter fault + attrib
INSTRUCTIONS :-
Enter attrib. values
Attribute: pedal
Choose value from list: hard, soft, Blocked, N/A
Enter new value: jammed

Entering new attribute condition (value)

Attribute: pedal
Choose value from list: hard, soft, Blocked, N/A
Enter new value: jammed

Screen after final attribute value has been entered

Fig. 2.12 Process for entering a new fault scenario

The task analysis illustrated in Figure 2.13 shows that the first procedure to be performed is the initialisation of the agent when various parameters are set. The agent would then go into a wait state where it looks out for messages to arrive at the BB server. Once a message has been received, the agent checks its origin. If the message is from another SA then it is considered to be a request to find a fault class. Therefore, SA will query the instance history using the IFCLEAR algorithm. If the message is from UA, then a check is performed to ascertain whether the same fault description is already contained within the instance history. If so, then the knowledge base is regenerated using the entire instance history as the reported fault has not been recognised by the expert system. This is because with some incremental algorithms the proportion of the instances in the instance history covered by the rule set might decrease as new instances are presented. In experiments, this has shown to be an issue with the RULES-4 algorithm when various user-defined parameters, such as the size of the short-term memory, were set inappropriately. This has led to several training instances subsequently failing to be covered by the generated rule set.

What happens next depends upon whether ISLE is being used. If it is not, a check is performed to see if the fault class has been provided. In case the class has not been specified, another agent or an engineer is asked to supply it.

If the fault class has been provided or ISLE is being used, then the format and training data files need to be generated. This depends upon the fault data. When there are new attributes, faults, or attribute values, then the format file for the algorithm needs to be regenerated, and the legal value file is also updated to take these changes into consideration. Once this has been done, a rule generation algorithm would be activated and a new rule set generated. The algorithm could be either RULES-4 or ISLE, or with appropriate modifications an algorithm such as ID3 could be used. The old knowledge base is deleted and replaced by the newly generated one. A blank HTML template file is created if necessary, as for example when new fault types have been added to the knowledge base. The agent then returns to the wait-state awaiting any new messages.

Architecture

SA is more like a conventional agent than UA, in that it is required to perform more complex tasks, such as reasoning about and processing the fault data and also sending and handling requests. This results in the need to pay greater attention to architectural details.

Architectural Requirements

Of the various architectural approaches available, a semi-reactive or partially hybrid approach was chosen. Several factors influenced this decision. Deliberative architectures are typically based upon BDI models of the agent and its environment that directly influence the decision-making processes of the agent. These models contain *inter alia* an expression of the beliefs of the agent with respect to the environment.

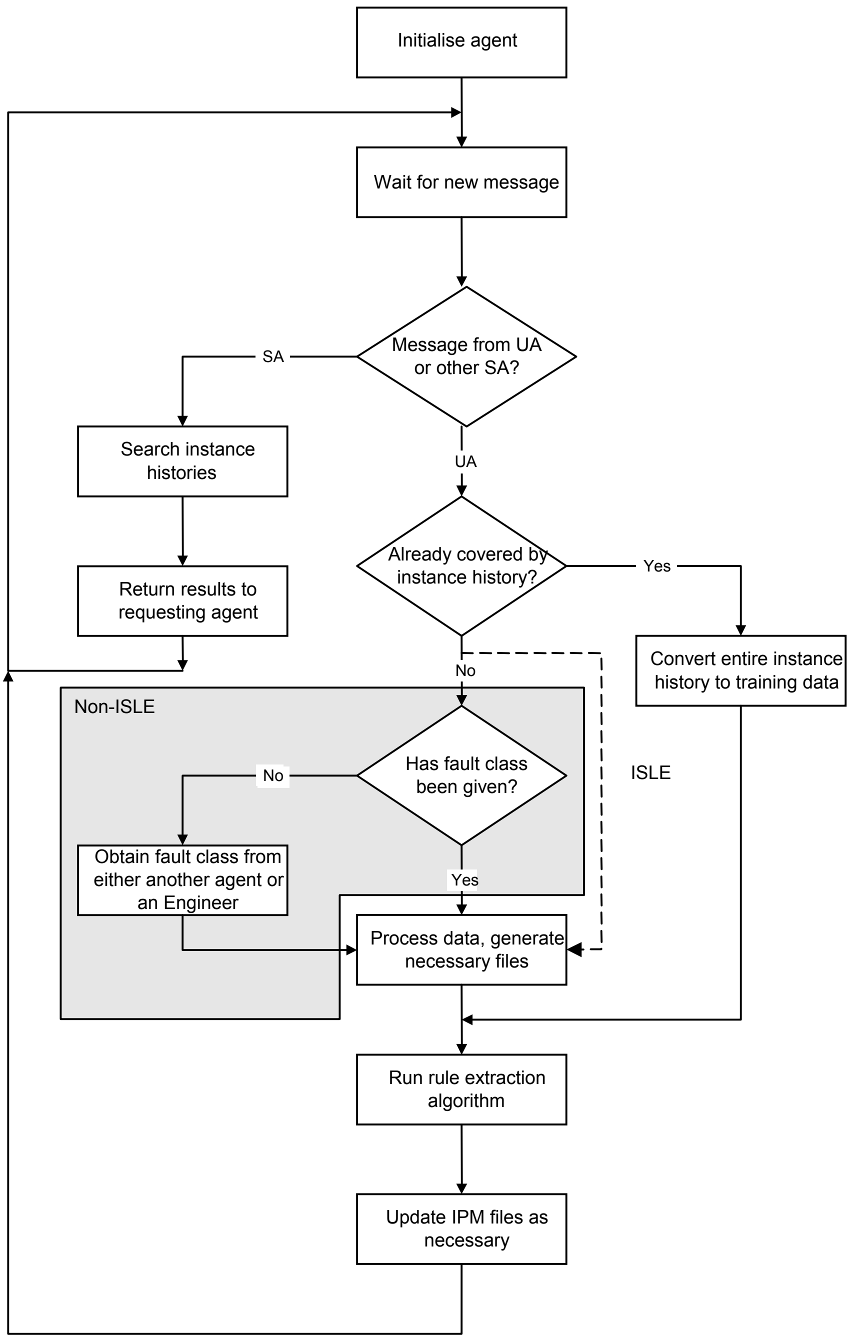

Initialise agent
Wait for new message
Message from UA or other SA?
SA
UA
Search instance histories
Return results to requesting agent
Already covered by instance history?
Yes
No
Convert entire instance history to training data
Non-ISLE
ISLE
Has fault class been given?
No
Yes
Obtain fault class from either another agent or an Engineer
Process data, generate necessary files
Run rule extraction algorithm
Update IPM files as necessary

Fig. 2.13 Task analysis for server agent

In this type of application, such models may be inappropriate. Within the proposed system, there is a large and potentially continually expanding population of UAs for which models might have to be maintained. To do so, symbolic manipulation would have to be performed. The interactions with the group of UAs as a whole will be frequent. However, for a single UA, it is probable that interaction would be relatively infrequent. This would raise issues regarding how to maintain accurate models of agents with which there is minimal interaction. As a deliberative architecture is likely to decide what actions to take based upon some model, an empirically constructed model may not be suitable for this application. For example, if, on the few occasions it has interacted with SA, a given UA has failed to give a fault class, this could potentially lead SA to assume, incorrectly, that this would hold true for the next interaction.

There are also situations where no *a priori* knowledge exists of what will occur. For example, the result of a help request to another SA cannot be anticipated. Therefore, there would be no real value in the agent attempting to create plans depending on possible outcomes of processes that have yet to occur. It is more logical for SA to base its decision-making mainly upon the state of the environment at the current instant. The agent should take decisions according to the results of its own previous actions or those of others. This leads to the conclusion that a reactive architecture might be the most appropriate, as reactive agents can respond directly to stimuli.

Despite the assertion that a reactive architecture appears to be the most suitable, there is still a need to maintain some form of database of other SAs. Contained within this database are the lists of other agents within the system and the scores relating to the similarity of the instance histories maintained. This ensures that an SA knows what assistance it may be able to obtain from other SAs. This database is not a model due to its being fairly static and not including any representation of how useful an agent has been in the past.

Specification of Architecture

There are different ways in which such a semi-reactive architecture could be implemented. It could be modelled on a rule-based system or a neural network. However, it was decided to investigate the application of the concepts of FSA as the basis for the architecture. An FSA is a model of a dynamic system that switches from one state to another depending on the current state and the result of the actions taken in that state.

The architecture presented here can be considered to have two main sources of inspiration and influence, namely situated automata (Kaelbling, 1991; Rosenschein and Kaelbling, 1995) and subsumption and behaviour-based architectures (Brooks, 1986, 1991; Maes, 1989).

Work has been conducted relating to situated automata in the domain of robotics where such systems are expressed as fixed sequential circuits (Kaelbling, 1991; Rosenschein and Kaelbling, 1995). A situated-automata-based system uses a synchronous approach, much in the way a digital circuit operates. This approach is concerned with the agent and its sensing and effecting and the formal logical specification of actions in an environment to perform a task (Kaelbling, 1991; Rosenschein and Kaelbling, 1995). This approach is not the most appropriate for the implementation of SA. Instead, the proposed FSA architecture is more concerned with the overall performance of a task and with the two environments that exist, the internal environment of the agent and the external world within which it operates.

The architectural approach presented here has some similarities with the subsumption and behaviour-based architectures in Brooks (1986, 1991) and also with the agent network architecture (Maes, 1989). Both of these architectures consist of modules that have certain behaviours prescribed to them. The modules then compete against one another to be activated and allowed to perform their action. The states within the FSA could be considered to be analogous to these various modules although they do operate on a much more basic level.

The FSA-based reactive agent architecture can in fact be considered analogous to a rule-based system. This is because theoretically the structure of the FSA could be represented as a series of rules. However, the use of an FSA simplifies the implementation in that it removes the requirement for a rule base. This is because a simple state table can be used to represent the FSA. Therefore there is no requirement for any inference engine. Instead the system queries the state table and activates the appropriate process. The use of a state table offers a benefit in that the table can be readily altered to allow the functionality and behaviour of the agent to be modified as required. The state table of the FSA depicted in Figure 2.14 is given in Table 2.1. The table was directly obtained from the FSA.

To illustrate the ease of changing the behaviour of SA, consider a hypothetical scenario where a report it just received is already covered in the instance history (see Figure 2.14). Instead of switching to generating a new knowledge base, suppose SA is required to change to the "check fault class" state. Only one entry in the state table need be altered, i.e. entry 6 for which the new state would now be "check fault class" instead of "generate KB". If required, this table could be altered dynamically, much in the way that links in the agent network in Maes (1991) can be changed.

In summary, a FSA lends itself to being used to implement SA in that it provides a system with reactive features while operating within a strictly bounded framework. It can also exhibit a complex functionality arising from states that perform simple tasks.

The architecture shown in Figure 2.14 illustrates the FSA at a high level of abstraction. As such, it contains several multiple process states. Within these are what could be considered as sub-FSAs that together constitute the entire FSA.

The concise representation of Figure 2.14 hides a great deal of the complexity of the system. By comparing the task analysis in Figure 2.13 and the FSA in Figure 2.14, it can be seen that the task and FSA are closely related. If the task analysis and full FSA were to be compared, it would become apparent that the FSA breaks down the sections of the task analysis into much smaller pieces. These separate states can be considered to represent very simple modules or elements.

However, through combining these simple elements a complex information-processing architecture can be obtained, achieving an outcome that is similar in parts to the concepts for reactive architectures outlined in Maes (1991).

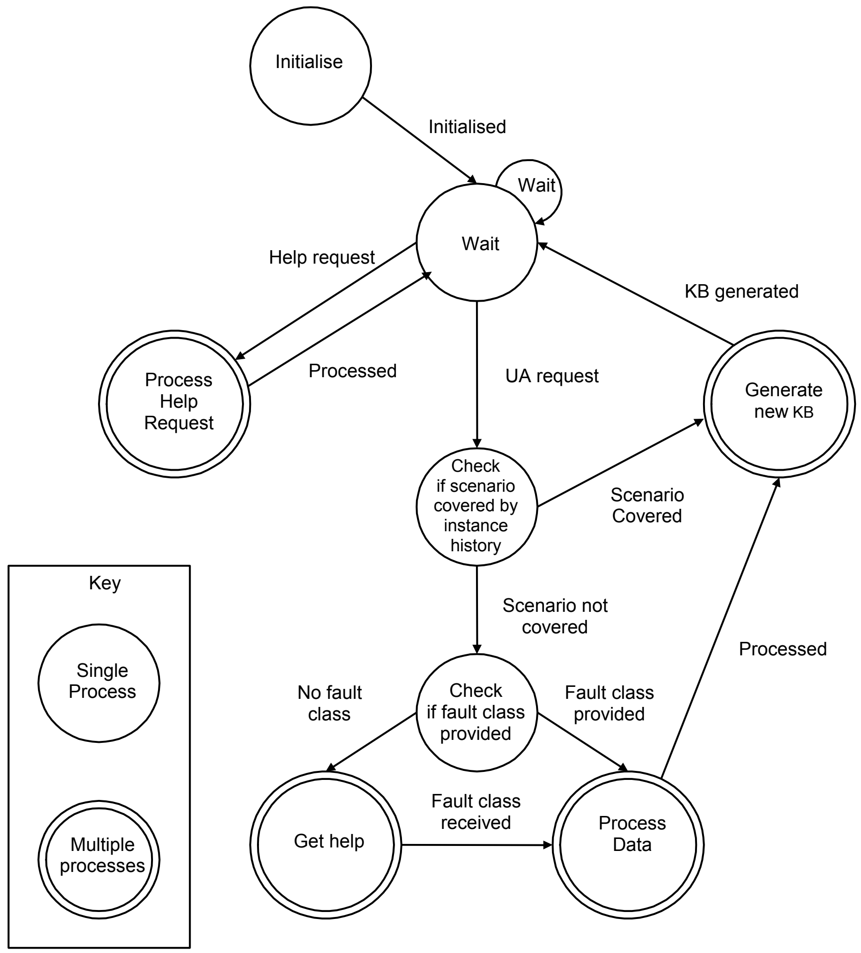


Fig. 2.14 FSA representing architecture of SA (*KB* = knowledge base)

Table 2.1 State table for FSA

	Current state	Result	New state		Current	Result	New
1	Initialise	Initialised	Wait		1	1	2
2	Wait	Wait	Wait		2	2	2
3	Wait	Help request	Process help	⇒	2	3	7
4	Wait	UA request	Check scenario		2	4	4
5	Process help	Processed	Wait		3	5	2

Table 2.1 (continued)

	Current state	Result	New state	Current	Result	New
6	Check scenario	Covered	Generate KB	4	6	8
7	Check scenario	Not covered	Check fault class	4	7	5
8	Check fault class	No fault class	Get help	⇒ 5	8	6
9	Check fault class	Fault class	Process data	5	9	7
10	Get help	Received	Process data	6	10	7
11	Process data	Processed	Generate KB	7	11	8
12	Generate KB	Generated	Wait	8	12	2

Operation

For a server-based system to operate, several software applications must be running at all times. First, Java's RMI registry must continually perform basic housekeeping tasks such as TCP/IP socket management for the Java virtual machines, which are the environments within which the applications run. Second, the BB server must be functioning so that UA is able to communicate with SA. Finally, SA must be running as it performs the various information processing tasks.

The user interface of SA is minimal as the user only interacts with SA, when a fault class has to be provided. Therefore the only output provided by SA is related to what task it is currently performing, so that the engineer is aware of its state. Figure 2.15 does not show the process of setting up the state table. To do this, the table is loaded into a custom Java object called *stateTable*. This object then stores the state table and provides methods for the SA to examine it as required.

The following list describes the purpose of and tasks performed by each normal and multiprocess state depicted within the FSA.

- *Initialise*: the agent is initialised. The list of legal fault types, attributes and values is loaded and other basic housekeeping tasks are performed.
- *Wait*: SA will wait in this state until a fault report is received by the BB server from a UA or until a help request is intercepted from another agent. A few other minor tasks are performed in this state. For example, if it is not already empty, the training data file will be cleared. Also if deemed appropriate the new fault can be checked by an engineer to ensure it is a genuine fault.
- *Process help request*: this is a multiple process state. SA will check the instance history using IFCLEAR and then, depending upon the outcome, it will either post a “can't help” message to the requesting agent or extract and send the fault class to the latter.
- *Check scenario*: SA checks if the new fault report is already contained inside the instance history maintained within it.
- *Check fault class*: SA will check whether the fault class for the new fault report has been specified. Note that if a pure ISLE-based approach is used, this step would immediately return the result that there is a fault class.

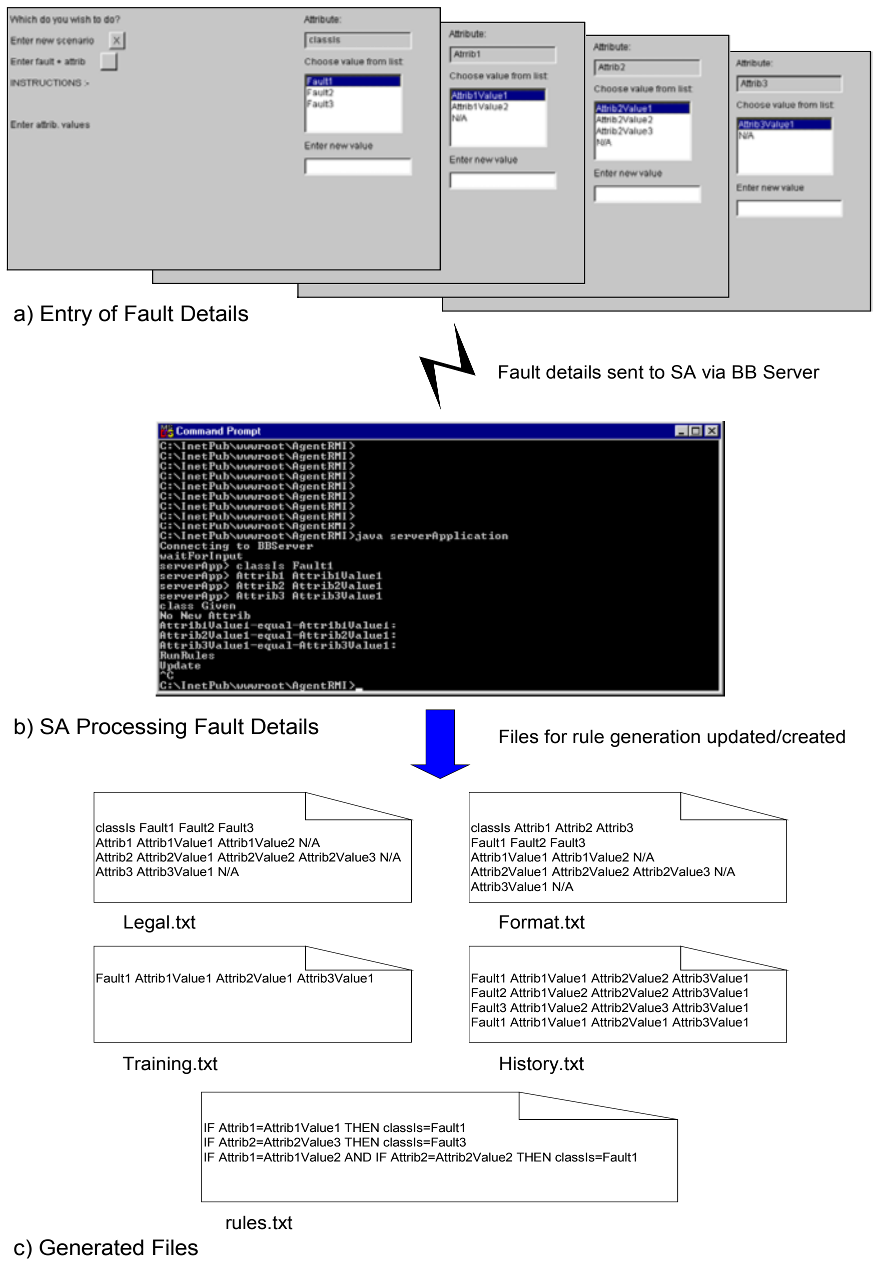
- *Get help* (a multiple process state): SA will check if there are any agents that can assist it. It does this by accessing its list of agents. If there are suitable agents then a help request will be sent and subsequently the reply will be processed. If there are no agents that can help or no other agents exist, then assistance is sought from the user.
- *Process data* (a multiple process state): the new fault report is processed and, if necessary, the format, legal value and instance history files are updated. If new attribute values or fault types have been added, then the legal value and format files are updated to take them into account. When a report contains a new attribute and its associated values, then the current instance history is modified to include this new attribute. The value N/A (not available) is given for that attribute for all of the existing instances. The instance history in that situation will be copied into the training data file. The format and legal value files will also be modified accordingly.
- *Generate new KB* (a multiple process state): the new fault report is appended to the instance history and the training data file. In situations where a new fault is same as one in the instance history (as mentioned previously) the contents of the instance history are copied into the training data file first. The rule extraction algorithm is executed and the IPM is updated as necessary. If a new fault has been specified then a template HTML file is generated so that the appropriate repair procedure can be inserted within it, and any files that require updating are duly modified.

2.3.3 Testing

The agent-based system proposed in this chapter consists of several distributed agents and applications that are interdependent. To assess the operation of the system, different issues have to be examined concerning:

- the correct transfer of data from UA to SA;
- the appropriate updating of files in SA;
- the correctness of the files; and
- the accuracy of the rule set generated.

An example is shown in Figure 2.15 where a new description for a known fault is given.



Attribute: classIs — Fault1, Fault2, Fault3
Attribute: Attrib1 — Attrib1Value1, Attrib1Value2, N/A
Attribute: Attrib2 — Attrib2Value1, Attrib2Value2, Attrib2Value3, N/A
Attribute: Attrib3 — Attrib3Value1, N/A

```
Command Prompt
C:\InetPub\wwwroot\AgentRMI>
C:\InetPub\wwwroot\AgentRMI>
C:\InetPub\wwwroot\AgentRMI>
C:\InetPub\wwwroot\AgentRMI>
C:\InetPub\wwwroot\AgentRMI>
C:\InetPub\wwwroot\AgentRMI>
C:\InetPub\wwwroot\AgentRMI>
C:\InetPub\wwwroot\AgentRMI>
C:\InetPub\wwwroot\AgentRMI>
C:\InetPub\wwwroot\AgentRMI>java serverApplication
Connecting to BBServer
waitForInput
serverApp> classIs Fault1
serverApp> Attrib1 Attrib1Value1
serverApp> Attrib2 Attrib2Value1
serverApp> Attrib3 Attrib3Value1
class Given
No New Attrib
Attrib1Value1-equal-Attrib1Value1:
Attrib2Value1-equal-Attrib2Value1:
Attrib3Value1-equal-Attrib3Value1:
RunRules
Update
^C
C:\InetPub\wwwroot\AgentRMI>_
```

```
classIs Fault1 Fault2 Fault3
Attrib1 Attrib1Value1 Attrib1Value2 N/A
Attrib2 Attrib2Value1 Attrib2Value2 Attrib2Value3 N/A
Attrib3 Attrib3Value1 N/A
```

```
classIs Attrib1 Attrib2 Attrib3
Fault1 Fault2 Fault3
Attrib1Value1 Attrib1Value2 N/A
Attrib2Value1 Attrib2Value2 Attrib2Value3 N/A
Attrib3Value1 N/A
```

```
Fault1 Attrib1Value1 Attrib2Value1 Attrib3Value1
```

```
Fault1 Attrib1Value1 Attrib2Value2 Attrib3Value1
Fault2 Attrib1Value2 Attrib2Value2 Attrib3Value1
Fault3 Attrib1Value2 Attrib2Value3 Attrib3Value1
Fault1 Attrib1Value1 Attrib2Value1 Attrib3Value1
```

```
IF Attrib1=Attrib1Value1 THEN classIs=Fault1
IF Attrib2=Attrib2Value3 THEN classIs=Fault3
IF Attrib1=Attrib1Value2 AND IF Attrib2=Attrib2Value2 THEN classIs=Fault1
```

Fig. 2.15 Illustrated example of entry and processing of new fault details with generated files

To provide an overview of how the agents within the system operate the stages of fault entry and processing are depicted. Initially, the new fault report is entered via the user interface of UA. Once this has been completed, the information is forwarded to the BB server, which then passes it to SA for processing. SA then generates the various files illustrated in Figure 2.15.

Testing showed that the system was able to cope with a variety of situations regarding data. Including scenarios where: new description of an existing fault is provided that also contains a new value for an existing attribute; a new type of fault is entered that contains existing attributes and utilises the existing values for those attributes; a fault occurs that requires both a new fault type and new attributes; an existing fault is given a new attribute and new attribute values.

2.4 Conclusions

The system described here has shown itself to be capable of accepting various types of fault reports and of processing these reports so that new knowledge bases can be generated. The system has also demonstrated a degree of robustness in being able to handle asynchronously submitted reports. This is due to its distributed nature, in particular, the use of the BB server to act as a depot for messages. This shows that agent-based systems together with Internet-based technologies can prove to be a vital tool for an enterprise when dealing with volumes of information that require processing.

References

Agre PE and Chapman D (1987) Pengi: an implementation of a theory of activity. In: Proceedings of the 6th national conference on artificial intelligence AAAI'87 Seattle, WA, USA, 1:268– 272

Aksoy MS (1993) New algorithms for machine learning. PhD. thesis, Cardiff University, UK

Al-Attar A (1991) Rule induction from mythology to methodology. Research and developments in expert systems VIII, London, UK, 85–103

Alty JL (1997) Agents and intelligent user interfaces. In: Proceedings of UNICOM tutorial on agents and intelligent user interfaces, Heathrow, London, UK

Alty JL, Griffiths D, Jennings NR et al. (1994) ADEPT-advanced decision environment for process tasks: overview and architecture. In: Proceedings of the 14th annual BCS technical conference on expert systems, 359–371, Cambridge, UK

Andreoli JM, Pacull F, Pareschi R (1997) XPECT: a framework for electronic commerce. IEEE Internet Comput., 1(4):40–48

Axelrod R (1984) The evolution of cooperation. New York: Basic Books.

Bond AH, Gasser L (1988) An analysis of problems and research in DAI. In: Bond AH and Gasser L (eds) Readings in distributed artificial intelligence, Morgan Kaufmann, San Mateo

Bratman ME, Israel DJ, Pollack ME (1988) Plans and resource-bounded practical reasoning. Comput. Intell., 4:349–355

Brooks R (1986) A robust layered control system for a mobile robot. IEEE Trans. Rob. Auto., 2:14–23

Brooks R (1991) Integrated systems based on behaviours. ACM SIGART Bulletin, 2(4):46–49

Chapman D (1987) Planning for conjunctive goals. Artif. Intell., 32:333–378

Cheng J, Fayyad UM, Irani KB et al. (1988) Improved decision trees: a generalised version of ID3. In: Proceedings of the 5th international conference on machine learning, Ann Arbor, USA, 100–106

Clark P, Boswell R (1991) Rule induction with CN2: some recent improvement. In: Proceedings of the 5th European conference on machine learning EWSL-91, 151–163

Clark P, Niblett T (1989) The CN2 inductive algorithm. Machine Learning, 3:261–283

Craig I (1995) Blackboard systems. Ablex Publishing Corporation, Norwood

Davies NJ, Weeks R, Revett MC (1997) Information agents for the world wide web. In: Nwana HS, Azarmi (eds) Software agents and soft computing: concepts and applications, Springer, Berlin, 81–89

Dreper BA, Collins RT, Brolio J et al. (1988) Issues in the development of a blackboard-based schema for image understanding. In Englemore RS, Morgan AJ (eds) Blackboard systems, Addison-Wesley, Wokingham, 189–218

Ekdahl B, Astor E, Davidsson P (1995) Toward anticipatory agents. In: Wooldridge M, Jennings NR (eds), Lect. Notes in Artif. Intell., 890:191–202

Englemore RS, Morgan AJ (1988) Blackboard systems. Addison-Wesley, Wokingham

Erman LD, Hayes-Roth F, Lesser VR et al. (1988) The hearsay II speech understanding system. In: Englemore RS, Morgan AJ (eds) Blackboard systems. Addison-Wesley, Wokingham, 31–86

Etzioni O, Weld D (1994) A softbot-based interface to the internet. Communications of the ACM, 37(7):72–76

Ferguson IA (1995) Integrated control and coordinated behaviour a case for agent models. In : Wooldridge M, Jennings NR (eds) Lect. Notes in Artif. Intell., 890:203–218

FIPA (1997) Foundation for intelligent physical agents. Agent communication language specification. [http://www.fipa.org/spec/f8a22.zip] FIPA Secretariat c/o S.I.A.–C.P. 3176, Strada Antica di Collegno 253, I-10146, Torino, Italy

Fisher M (1995) Representing and executing agent-based systems. In: Wooldridge M, Jennings NR. (eds) Lect. Notes in Artif. Intell., 890:203-218

Foner L, Crabtree IB (1996) Multiagent matchmaking. B.T. Technol. J. 14(4):115–123

Forsyth R (1989) Machine Learning: Principles and Techniques. Chapman and Hall, London

Franklin S, Graesser L (1996) Is it an agent, or just a program?: A taxonomy for autonomous agents. In: Muller JP, Wooldridge MJ, Jennings NR, (eds) Lect. Notes in Artif. Intell., 1193: 21–35

Franklin S, Graesser A, Olde B et al. (1996) Virtual mattie–an intelligent clerical agent. [http://www.msci.memphis.edu/~franklin/Vmattie.html] Department of Mathematical Sciences, the University of Memphis, Memphis, Tennessee 38152, USA

Genesereth MR, Nilsson NJ (1988) Logical foundations of artificial intelligence. Morgan Kaufmann, San Mateo

Gensereth MR, Fikes RE (1992) Knowledge interchange format, Version 3.0 Reference Manual. Stanford University Report Logic-92-1, Stanford University, Stanford, CA 94305-9025, USA

Genesereth MR, Ketchpel SP (1994) Software agents. Communications of the ACM 37(7):48–53

Georgeff M (1988) Communication and interaction in multiagent planning. In: Bond AH, Gasser L (eds) Readings in distributed artificial intelligence, Morgan Kaufmann, San Mateo, 200–204

Georgeff MP, Lansky AL (1987) Reactive reasoning and planning. In: Proceedings of the 6th National Conference on Artificial Intelligence (AAAI-87), pp. 677-682, Seattle, WA, USA

Gibbins PF, Kearney P, Sehmi A et al. (1994) Intelligent agents in multimedia systems. Sharp Tech. J., 60:11–14

Gray R, Kotz D, Nog S et al. (1996) Mobile agents for mobile computing. Technical Report PCS-TR96-285, Dartmouth College [ftp.cs.dartmouth.edu/TR/TR96-285.ps.Z], Department of Computer Science, Dartmouth College, 6211 Sudikoff Laboratory, Hanover, NH 03755-3510, USA

Hancox PJ, Mills WJ, Reid BJ (1990) Keyguide to information sources in artificial intelligence/ expert. In: Lawrence KS: Ergosyst

Hayes-Roth B (1993) An architecture for adaptive aystems. Stanford University report number STAN-CS-TR-93-1496 (also available as KSL 93-19), Computer Science Dept, 353 Serra Mall, Stanford University, Stanford CA 94305–9025, USA

Hayes-Roth B, Gaba D, Uckun S et al. (1994) Guardian: a prototype intelligent agent for intensive care monitoring. In: Proceedings of the 12th national conference on artificial intelligence, 2:1503

Hewitt C (1977) Viewing control structures as patterns of passing messages. Artif., Intell. 8(3):323–364

Holt J, Rodd MG (1994) An architecture for real-time distributed artificial intelligent systems. . Real-Time Systems, 6:263–288

Huang J, Jennings NR, Fox J (1995) An architecture for distributed medical care. In: Wooldridge M, Jennings N.R., (eds) Lect. Notes in Artif. Intell., 890:219–232

Huhns M.N. (1987) (ed) Distributed artificial intelligence. Pitman, London

Huhns MN, Jacobs N, Ksiezyk T et al. (1992) Integrating enterprise information models in carnot. In: Proceedings of the international conference on intelligent and cooperative information systems, Rotterdam, Netherlands, 32–42

Hunt EB, Marin J, Stone PJ (1966) Experiments in induction. Academic press, New York, USA

Ito A, Yano H (1995) Can machines ever learn to cooperate? The emergence of cooperation in a society of autonomous agents. J. Comm. Res. Lab., 42(3):243–249

Jennings NR (1992) Using GRATE to build cooperating agents for industrial control. IFAC symposium on artificial intelligence in real-time control, Delft, Netherlands, 1–6

Jennings NR (1995) Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. Artif. Intell., 75:195–240

Jennings NR, Faratin P, Norman TJ et al. (1996) ADEPT: Managing business processes using intelligent agents. In: Proceedings BCS expert systems 96 conference, Cambridge, 5–23

Jennings NR, Mamdani EH, Laresgoiti I et al. (1992) GRATE: a general framework for cooperative problem solving. IEE-BCS J. Intell. Sys. Eng., 1(2):102–114

Jennings NR, Varga LZ, Aarnts RP et al. (1993) Transforming standalone expert systems into a community of co-operating agents. Eng. App. Artif. Intell., 6(4):317-331

Jennings NR, Wooldridge M (1995) Applying agent technology. App. Artif. Intell., 9(4):357–369

Jennings NR, Wooldridge M (1998) Application of intelligent agents. In: Jennings NR, Wooldridge M (eds) Agent technology: foundations, applications and markets, Springer-Verlag, Berlin, 3–28

Kaelbling LP (1991) A situated-automata approach to the design of embedded agents. ACM SIGART Bulletin, 2(4):85–88

Kato K, Someya Y, Matsubara K et al. (1999) An approach to mobile software robots for the www. IEEE Trans. Know. Data Eng. , 11(4):51–548

Kearney P (1996) Personal electronics: personal agents. In: Proceedings 1st IEE colloquium on intelligent agents 7/1–7/6

Khedro T, Genesereth MR (1994) Progressive negotiation for resolving conflicts among distributed heterogeneous co-operating agents. In: Proceedings of the 12th national conference on artificial intelligence, Seattle, USA, 1:381–386

Knight K (1993) Are many reactive agents better than a few deliberative ones? In: Proceedings of the 13th international joint conference on artificial intelligence (IJCAI'93), Chambery, France, 432–437

Kodratoff Y (1988) Introduction to machine learning. Pitman Publishing, London

Kowalski R, Sadri R (1996) Towards a unified agent architecture that combines rationality with reactivity. In: Proceedings of the international workshop logic in databases LID, San Miniato, 137–149

Kozierok R, Maes P (1993) A learning agent for scheduling meetings. In: Proceedings ACM-SIGCHI international workshop, Orlando, USA, 81–88

Kumar D, Shapiro SC (1991) Architecture of an intelligent agent in SNePS. ACM SIGART Bulletin, 2(4):89–92

Labrou Y, Finin T (1997) A proposal for a new KQML specification. The University of Maryland, Baltimore County report TR CS-97–03, The University of Maryland, Baltimore County, 1000 Hilltop Circle, Baltimore, MD21250, USA

Lander SE, Lesser VR, Connell ME (1990) Knowledge-based conflict resolution for cooperation among expert agents. In: Proceedings of MIJ-JSME workshop, 253–268

Lashkari Y, Metral M, Maes P (1994) Collaborative interface agents. In: Proceedings of the 12th national conference on artificial intelligence AAAI'94, Seattle, USA, 1:444–449

Lieberman H (1995) Letizia: an agent that assists web browsing. In: Proceedings of the 14th international conference on artificial intelligence IJCAI'95, Montreal, Canada, pp. 924–929

Lieberman H (1997) Autonomous interface agents. In: Proceedings of the ACM conference on computers and human interfaces CHI-97, Atlanta, USA

Lieberman H (1999) Personal assistants for the web: A MIT perspective. In Klusch M (ed): Intelligent information agents: agent-based information discovery and management on the Internet, Springer, Berlin, Germany, pp. 279–292

Maes P (1989) The dynamics of action selection. In: Proceedings of the 11th international conference on artificial intelligence IJCAI-89, Detroit, USA, 2:991–997

Maes P (1991) Designing autonomous agents: theory and practice, from biology to engineering and back. MIT Press, Cambridge, USA

Maes P (1994) Agents that reduce work and information overload. Communications of the ACM, 37(7):30–40

Maes P (1995) Modelling adaptive autonomous agents. In: Langton CG (ed) Artificial life, MIT Press, Cambridge, USA

Mamdani EH, Charlton PM (1996) Agent-based support for public-information kiosks. In: Proceedings of the 1st IEE colloquium on intelligent agents, London, UK, 4/1–4/3

Mayfield J, Labrou Y, Finin T (1996) Evaluation of KQML as an agent communication language. IJCAI'95 workshop, Montreal, Canada, 347–360

Meghini C (1997) A reactive logical agent. In: Proceedings of the 1st international workshop on cooperative information agents CIA'97, Kiel, 148–158

Michalski RS (1983) A theory and methodology of inductive learning. In: Michalski RS, Carbonell JG, Mitchell TM (eds) Machine learning–an artificial intelligence approach, Morgan Kaufmann, Los Altos, 83–134

Michalski RS, Mozetic I, Hong J et al. (1986) The multi-purpose incremental learning algorithm AQ15 and its testing, application in three medical domains. In: Proceedings of the 5th national conference on artificial intelligence, Philadelphia, USA, 1041–1047

Moffat D, Frijda NH (1995) Where there's a will there's an agent. In: Wooldridge M, Jennings NR (eds) Lect. Notes in Artif. Intell., 890:245–260

Muller JP (1996) The design of intelligent agents: A layered approach. Lect. Notes in Artif. Intell., 1177

Muller JP, Pischel M, Thiel M (1995) Modelling reactive behaviour in vertically layered agent architectures. In: Wooldridge M, Jennings NR (eds) Lect. Notes Artif. Intell.,: 890: 261–276

Nwana HS, Ndumu DT (1996) An introduction to agent technology. British Telecom Technology J., 14(2):55–67

O'Brien P, Wiegand M (1997) Agents of change in business process management. In: Nwana HS, Azarmi N (eds) Software agents and soft computing, concepts and applications, Springer-Verlag, Berlin, Germany, pp. 132–245

Osawa E-I, Tokoro M (1992) Collaborative plan construction for multi-agent mutual planning. In: Werener E, Demazeau Y (eds), Decentralized AI3, Elsevier, Oxford, USA, pp. 169–185

Papazoglou M, van den Heuvel WJAM (2000) Configurable business objects for building evolving enterprise models and applications. In: van der Aalst WMP, Desel J, Oberweis A (eds), Business process management: models, techniques and empirical studies, Lect. Notes Comp. Sci. N. 1806, pp. 328–344

Pham DT, Aksoy MS (1993) An algorithm for automatic rule induction. Artif. Intell. Eng., 8:277–282

Pham DT, Aksoy MS (1995) RULES: a simple rule extraction algorithm. Expert Sys. App. 8(1):59–65

Pham DT, Dimov SS (1997a) An efficient algorithm for automatic knowledge acquisition. Pattern Recognition 30(7):1137–1143

Pham DT, Dimov SS (1997b) An algorithm for incremental inductive learning. In: Proceedings of the Institution of Mechanical Engineers Part B: J. Eng. Manuf., 211:239–249

Pham DT, Dimov SS, Setchi RM (1998) Intelligent product manuals. In: Proceedings of the Institution of Mechanical Engineers Part B: J. Eng. Manuf., 213:65–76

Pham DT, Dimov SS, Soroka AJ (2001) Knowledge acquisition for intelligent product manuals. In: Proceedings of the Institution of Mechanical Engineers Part B: J. Eng. Manuf., 215:97–103

Pham DT, Soroka AJ (2005a) ISLE – A novel immune-system inspired rule extraction algorithm. In: Proceedings of the 16th IFAC World Congress, Prague

Pham DT, Soroka AJ (2005b) RULES_IS immune-network inspired machine learning algorithm. In: Proceedings of the 1st I*PROMS virtual conference on intelligent production machines and systems

Pham DT, Soroka AJ (2006) An algorithm based on the immune system for extracting fault classes from instance histories. In: Proceedings of the 2nd I*PROMS virtual conference on intelligent production machines and systems, pp. 313–318

Quinlan JR (1986) Induction of decision trees. Machine learning Vol. 1. Kluwer Academic Publishers, Boston, USA, 81–106

Quinlan JR (1987) Generating production rules from decision trees. In: Proceedings of the 10th international joint conference on artificial intelligence, Milan, Italy, pp. 304–307

Quinlan JR (1988) Induction knowledge and expert systems. In: Gero JS, Stanton R (eds) Artificial intelligence developments and applications, North Holland, Amsterdam, 253–271

Rao A, Georgeff MP (1995) BDI agent from theory to practise. In: Proceedings of the 1st international conference on multi-agent systems, San Francisco, USA, 312–319

Rosenschein SJ, Kaelbling LP (1995) A situated view of representation and control. Artif. Intell. 73(1-2):149–173

Schlimmer JC, Fisher D (1986) A case study of incremental concept induction. In: Proceedings of the 5th national conference on artificial intelligence, Philadelphia, USA, pp. 495–501

Schroeder M, Mora IA, Pereira LM (1996) A deliberative and reactive diagnosis agent-based on logic programming. In: Muller JP, Wooldridge M, Jennings NR (eds) Lect. Notes in Artif. Intell., 1193:293–307

Sen S, Sekaran M, Hale J (1994) Learning to co-ordinate without sharing information. In: Proceedings of the 12th national conference on artificial intelligence, Seattle, USA, 1:426–431

Sheth BD (1994) A learning approach to personalized information filtering. MSc Thesis: MIT, media laboratory communications and sponsor relations, 77 Massachusetts Avenue, Cambridge, Massachusetts, USA, pp. 2139–4307

Singh MP, Huhns MN (1999) Social abstractions for information agents. In: Klusch M (ed) Intelligent information agents: agent-based information discovery and management on the internet, Springer, Berlin, Germany, pp. 37–52

Struthers A (1996) Agent-based developments: a discussion of opportunities and hurdles, intelligent agents and their applications. In: Proceedings the 1st IEE colloquium on intelligent agents, London, UK, 6/1–6/3

Sycara KP (1989) Argumentation: planning other agents' plans. In: Proceedings of the 11th international joint conference on artificial intelligence, Detroit, USA, 2:517–523

Sycara KP (1999) In-context information management through adaptive collaboration of intelligent agents. In: Klusch M (ed) Intelligent information agents: agent-based information discovery and management on the internet, Springer, Berlin, Germany, pp. 78–99

Utgoff PE (1988) ID5: an incremental ID3. In: Proceedings of the 5th international conference on machine learning, Ann Arbor, Michigan, USA, pp. 107–120

Voorhees EM (1994) Software agents for information retrieval. In: Etzioni O (ed) Software, agents – spring symposium, AAAI Press, pp. 126–129

White JE (1994) Telescript technology: the foundations for the electronic market place, http://www.genmagic.com/telescript General Magic, 420 North Mary Ave., Sunnyvale, CA 94086, USA

Wooldridge MJ, Jennings NR (1995) Agent theories, architectures and languages: a survey. In: Wooldridge MJ, Jennings N. (eds) Lect. Notes in Artif. Intell., 890:1–39

Wooldridge MJ (1996a) Agents as a Rorschach test: a response to Franklin and Graesser. In: Muller JP, Wooldridge MJ, Jennings NR (eds) Lect. Notes in Artif. Intell., 1193: 47–48

Wooldridge MJ (1996b) What agents aren't: A discussion paper. In: Proceedings of the 1st IEE colloquium on intelligent agents, London, UK, 2/1–2/2

Zlotkin G, Rosenschein JS (1989) Negotiation and task sharing amongst autonomous agents in cooperative domains. In: Proceedings of the 11th international joint conference on artificial intelligence IJCAI'89, Detroit, USA, 2:912–917

Chapter 3
Multi-agent Simulation-based Decision Support System and Application in Networked Manufacturing Enterprises

H. Ding, W. Wang, M. Qiu and J. Dong

Abstract The recent financial crisis has had a major negative impact on the global economy, and particularly had a significant impact on the global manufacturing industry due to the ever-decreasing customer demand. For manufacturing enterprises, especially those who run businesses in multiple countries, it is now a good time to operate in a smarter way and lead the new era that is taking shape underneath the present crisis. We introduce a multi-agent-based simulation tool in this chapter, with a description of the overall architecture, modelling elements, operational policies, etc. The tool has been used in a commercial project with a leading high-tech manufacturer. The complex relationships between service levels, inventory cost, transportation cost, and forecasting accuracy were well studied. The project results show that networked enterprises can really get better insight from such a quantitative analysis and would be able to identify solid opportunities for cost saving and performance improvement.

Keywords Multi-agent, simulation, networked manufacturing enterprises

3.1 Introduction

The recent financial crisis has had a major negative impact on the global economy, and particularly had a heavy impact on the global manufacturing industry due to the ever-decreasing customer demand. Big crisis represents big opportunity in the meantime. For manufacturing enterprises, especially those who operate in multiple countries, it is now a good time to operate in a more intelligent way and lead the new era that is taking shape underneath the present crisis. The question is how? How to operate in a smarter way? How to perform well in the current difficult

H. Ding (✉), W. Wang, M. Qiu and J. Dong
Supply Chain Management Research Department, IBM Research-China, ZGC Software Park, Beijing 100193, P.R. China
e-mail: dinghw@cn.ibm.com

economic environment, not only achieving increased revenue, margin growth and improved operational excellence? No easy answer. However, there are a number of ways to improve the current strategy and operations by leveraging advanced analytical technologies. Agent-based simulation is one of the advanced and effective technologies for decision support.

With the development of economy globalization, manufacturing enterprises are no longer operating “siloed” plants, but networked enterprises tightly connected with upstream suppliers and downstream distributors. The production of raw materials, components and semi-products are pushed to upstream OEM (original equipment manufacturer) suppliers as much as possible, while the core manufacturer focuses only on those high-value-added and technology-intensive processes. The advantages of such networked manufacturers help reduce the total supply chain cost, and maximize the efficiency of each single process. But there is no free lunch in the world. Because of the existence of multiple echelons, which are responsible, for different tasks, and the fact that these tasks are interrelated, the management and coordination of such networked manufacturing enterprises present a considerable challenge. For example, demand forecasting for upstream suppliers is very challenging, because of the well-known bullwhip effects plus the volatile market demand. Another challenge is about how to fulfill customer demand with different service levels which achieves the best trade-off between supply chain cost and service level. A case study is presented in this chapter to showcase how agent-based simulation technology can help assess a networked manufacturing enterprise and identify opportunities for improvement.

In the remainder of this chapter, Sect. 3.2 gives a literature review in the field of simulation models, multi-agent simulation, plus the tools and applications using simulation techniques. In Sect. 3.3, the multi-agent simulation model and the simulation software’s architecture are presented. The simulation software is applied in a consulting project with a global manufacturing enterprise. Details related to the case study and simulation modelling are given in Sect. 3.4. Conclusions and perspectives are given in Sect. 3.5.

3.2 Literature Review

Simulation has been widely used in manufacturing enterprises to assist in strategic-level and/or operational-level decision-making. For example, Lendermann *et al.* (2003) demonstrated the use of simulation in semiconductor manufacturing. Towill *et al.* (1992) applied simulation techniques to evaluate the impact of different supply chain strategies on demand fluctuation. Tzafestas and Kapsiotis (1994) utilized a combined analytical and simulation model to analyse supply chains. Researchers also conducted many surveys on simulation methods and models from different perspectives. Kleijnen (2003) provided a survey of simulation applications in supply chain management and reviewed four types of simulation techniques, respectively spreadsheet simulation, system dynamics, discrete event simulation (DES), and business games. Terzi and Cavalieri (2004) provided a detailed review of the application of the high-level architecture (HLA) for manufacturing and supply chain simulation. They compared local simulation with parallel and distributed simulation paradigms. In this section we will review techniques such as local simulation, distributed simulation and multi-agent simulation, and especially tools and applications for decision support in manufacturing industries.

3.2.1 Simulation Methods

Normally simulation methods fall into two categories: local simulation and distributed simulation. Local simulation is popular in supply chain simulation area because of its simplicity. Among local simulation techniques, Monte-Carlo simulation (MCS) and DES are two most important methods. MCS is generally a static method, and usually being performed in spreadsheet based tools. MCS is light-weighted and very useful for creating some high-level models and getting preliminary results. Details and applications of MCS can be found in Rubinstein and Kroese (2007).

DES is heavier but in meantime it can handle more details with better capability to evaluate. It represents individual events and incorporates uncertainties. Banks *et al.* (2002) surveyed many DES studies in supply chain management. More details on DES can be found in Law and Kelton (2000). DES is already part of the MRP/ERP (material requirement planning/enterprise resource planning) toolbox to conduct quantitative analysis of costs or benefits from strategic, tactical and operational policies.

Distributed simulation has advantages on leveraging computation resources from multiple machines to handle larger problems within shorter time. Compared with sequential simulation distributed simulation requires little additional memory. Simulation of a system can be adapted to the structure of the available hardware. Literature on distributed simulation includes Sudra *et al.* (2000), Gan *et al.* (2000) and Brun *et al.* (2002).

3.2.2 Multi-agent Simulation

Multi-agent simulation, which has been researched since the early 1990s, is regarded as one of the most promising technologies for supply chain management (see Nissen, 2000; Swaminathan, 1997). The autonomous nature of independent agents is suitable for supply chain management where a single company cannot govern the whole supply chain coordination. The intelligence or behaviour of each agent can help the various planning activities in supply chains. Agents can also form dynamic collaboration network for turbulent supply chains through contracts or negotiations that are supported by agent interaction protocols.

The difficulty in applying multi-agent simulation technology rests in agent intelligence or behaviour modelling. Agents can interact or collaborate with each other or the environment via specific communication protocols for negotiation and contracts. Chen *et al.* (2000) described an approach that employs subcontract auctions to form dynamic and efficient supply chains. Fox *et al.* (2000) presented an integrated agent-based supply chain model, which can adapt to stochastic events such as breakdown of plants. Mondal and Tiwari (2003) suggested a mobile-agent system wherein mobile agents migrate to the systems of partner companies to find and evaluate potential partners for supply chain formation. Shen *et al.* (2003) analysed the advantages of agent technology, and propose a multi-agent architecture and an infrastructure for an Internet-enabled collaborative enterprise.

Reinforcement learning is a relatively new way to model agent behaviours in multi-agent simulation. Reinforcement learning is defined by Kaelbling *et al.* (1996) as "the problem faced by an agent that must learn behaviour through trial-and-error interactions with a dynamic environment". A reinforcement-learning-based approach is ideally suited for decision-making problems that are either Markov, or semi-Markov decision problems. The reinforcement learning approach is detailed by Sutton and Barto (1998).

In reinforcement learning the agents have learning/reasoning capabilities and have an explicit representation of the environment. Pontrandolfo *et al.* (2002) presented a global supply chain management framework utilizing reinforcement learning technique combined with Markov decision theory. Moreover, Qiu *et al.* (2007) adopted the reinforcement learning approach to model a distribution system with multiple agents.

3.2.3 Tools and Applications

A number of general-purpose simulation tools are available in the market for supply chain simulation, such as Arena, AnyLogic, AutoMod, etc. When using these tools, analysts usually create simulation models with a set of pre-defined logic elements, like CreateNode, DecisionNode, DisposeNode, etc. Advantages of using general-purpose simulation tools are that analysts can create a model much faster than using programming language and the model can be complex enough to cover real situations.

There are also some simulation tools designed specifically for manufacturing industries. IBM Research has developed "best-of breed" supply chain simulation tools including Supply Chain Analyzer (Bagchi *et al.*, 1998), which enjoyed considerable success both in the IBM internal supply chain and in external consulting. However, there is a need to build a next-generation supply chain modelling tool incorporating state of the art software development technology and bringing in enhanced modelling and simulation capability to the new tool. IBM China Research Lab developed a new supply chain modelling tool, SmartSCOR (Dong *et al.*, 2006), and recently its supply chain simulation part of the tool became a standalone tool named GBSE (General Business Simulation Environment) (see Wang *et al.*, 2008). It is an integrated supply chain simulation tool to help making tactical-level decisions. It supports industry-scale data sets and a rich set of supply chain processes. GBSE is based on the Eclipse platform and implemented as an Eclipse rich client platform.

A bunch of simulation libraries are also available, which can be leveraged when building simulation tools to avoid doing the work from scratch. Some famous and powerful libraries, such as Repast 2008 and Scalable Simulation Framework (SSF, 2008), are available via the Internet.

Applications of simulation for supply chain decision-making have also been well studied. Indeed, supply chains are made up of heterogeneous production subsystems gathered in vast dynamic and virtual coalitions. Intelligent distributed systems, e.g., multi-agent systems, enable increased autonomy of each member in the supply chain. Each agent in the simulation system pursues individual goals, while satisfying both local and external constraints. Therefore, one or several agents can be used to represent each partner in the supply chain such as a plant, distributor, or retailer.

DragonChain is an application implemented by Kimbrough *et al.* (2002) at the University of Pennsylvania (Philadelphia, PA, USA) to simulate supply chain management, and more particularly to reduce the bullwhip effect in the Beer Game. NetMan (NETworked MANufacturing) formalized networked organizations and production operations in order to obtain agile manufacturing networks in a dynamic environment (Cloutier *et al.*, 2001). Different than DragonChain, this multi-agent system manages an actual supply chain, rather than the Beer Game. Each company is cut in NetMan centres, i.e., independent, collaborating business units. The NetMan centres of a company, which coordinates each other. The coordination is based on contracts and conventions, which are formalized according to the convention, agreement, and transaction model. MASCOT (Multi-Agent Supply Chain cOordination Tool) is a reconfigurable, multilevel, agent-based architecture for planning and scheduling aimed in improving supply chain agility (Sadeh-Koniecpol *et al.*, 1999). It coordinates production among multiple (internal or external) facilities, and evaluates new product/subcomponent designs and strategic business decisions (e.g., make-or-buy or supplier selection decisions) with respect to capacity and material requirements across the whole supply chain.

3.3. Problem Description and Approach

For modelling and analysing both the internal processes and external interactions of a networked enterprise, we developed a discrete event simulation platform named Multi-Agent Supply Chain Simulator, or MASCS for short. MASCS is an Eclipse-based software tool programmed with Java. It supports the simulation of multi-period, multi-product, and multi-echelon supply chain models.

3.3.1 Platform Architecture

The MASCS platform comprises three major modules, respectively the discrete event simulation engine, MASCS model, and simulation cockpit, which are all based on the Eclipse platform and Java. Figure 3.1 depicts the platform architecture.

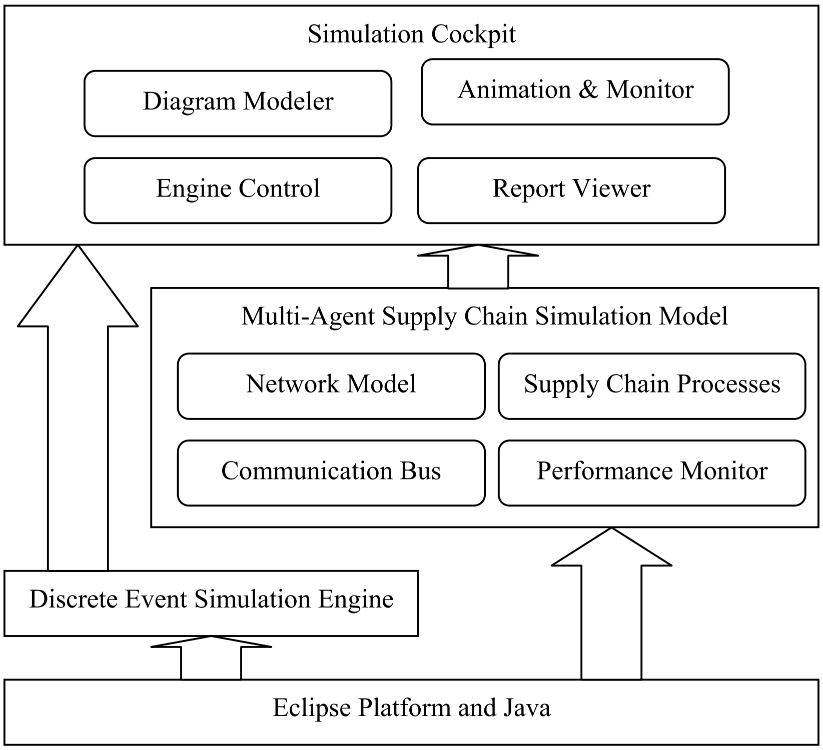


Fig. 3.1 Platform architecture

A Java-based discrete event simulation engine is embedded in the platform which triggers events and drives the processes of a supply chain model. The MASCS model is the core of the proposed platform. The model contains four major components. The network model represents the static structure of a supply chain, comprising a set of agents. Each agent represents a facility or a member in a supply chain. Supply chain processes are the dynamic part of a supply chain, including order processing, inventory control, procurement, manufacturing, transportation, etc. The communication bus is designed for the interactions between different agents in the model. It provides a unified method for the interactions of different types of agents. The performance monitor collects information from agents during the simulation process and calculates the corresponding key performance indicators. After each simulation run, the simulation reports are created accordingly.

The simulation cockpit is the interface for the end users, i.e. supply chain modellers and analysts. The diagram modeller is provided for creating the model in a drag-and-drop style. The animation and monitor component gives an intuitive view of simulation runs. Both animation and dynamic figures are supported. The engine control component is used to control the simulation engine, where users can also set simulation parameters like stopping time, warm-up period, animation switch on/off, etc.

3.3.2 Multi-agent Supply Chain Simulation Model

MASCS supports supply chain analysis with consideration of multi-period, multi-product, and multi-echelon. Four types of agents are modelled in the supply chain model: suppliers, manufacturers, DCs (distribution centres) and retailers. Figure 3.2 shows the network structure.

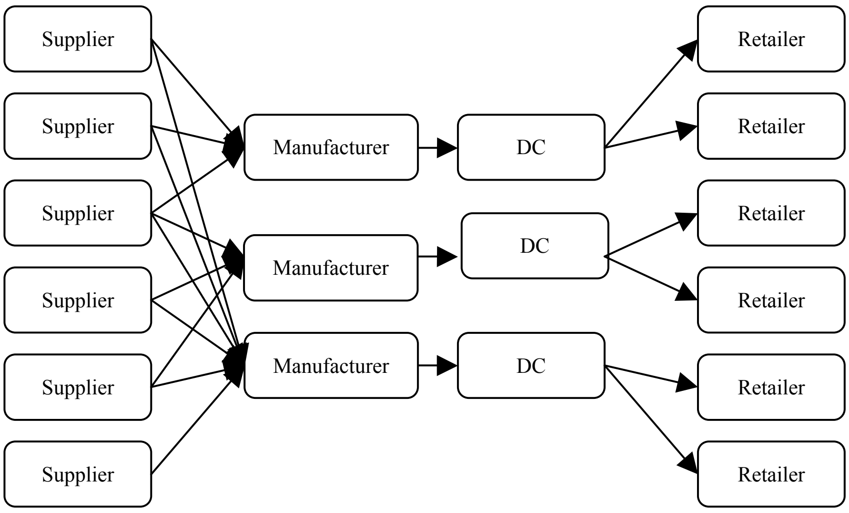


Fig. 3.2 A typical supply chain network structure

Manufacturers and DCs are key facilities in a supply chain. The two types of facilities are the focuses of the modelling and analysis. Manufacturers assemble raw materials and/or components to produce finished goods. Each manufacturer may contain multiple production lines. The tool user can choose sequential production or parallel production for production lines. Production is executed based on the definition of bill-of-material and other detailed processes. DCs receive finished goods from upstream plants and/or DCs, and deliver the finished goods to retailers. Different sourcing policies, inventory control policies and transportation policies can be applied in DCs. Manufacturers focus more on the production processes, including manufacturing order release, production planning and scheduling, bill-of-materials, etc. DCs focus more on inventory control and transportation processes, including inventory control policies, transportation policies, vehicle dispatch policies, etc.

Suppliers and retailers are usually used to represent external companies. Accordingly, no much-detailed processes are modelled in these facilities. Suppliers are the original sources of the materials in the entire supply chain, which provide raw materials to manufacturers and/or DCs. The internal processes are simplified since we intend to use these facilities to model external companies. Similar to suppliers, the internal processes of retailers are also simplified. Different retailers have different order intervals and requests for finished goods, and receive product delivery from upstream facilities. Some retailers may place orders each day, and some may place orders once a week. The order frequency is an input parameter.

3.4 Modelling and Analysis

We conducted a supply chain simulation study for a US based high-tech manufacturing company. This company is a typical networked manufacturer, which has global existence in multiple countries. It sources raw materials, components and semi-products from several hundreds of suppliers. Manufacturing is the core business for this company. It sells finished products to customers through direct shipment, or via channel vendors. For the sake of confidentiality, in this chapter we name it ABC Company or ABC.

As a multinational high-tech manufacturer, ABC's product portfolio is very complex. The bill-of-materials of finished products is very complicated, with hundreds of different components. ABC operates a global supply chain with suppliers and customers located in many different countries. ABC purchases raw materials from suppliers, assembles finished goods in plants, ships to DCs, and then delivers to retailers.

The objective of this study is to evaluate ABC's supply chain performance, and identify opportunities for potential improvements. In this study, the ABC supply chain is measured from three aspects: total supply chain cost, average service level, and total carbon emission.

3.4.1 Baseline Simulation Model

The baseline model represents the as-is status of the ABC supply chain. A multi-period, multi-product, and multi-echelon supply chain model is created according to the company's supply chain structure and business operations.

The simulation time horizon of the model is set to be 1 year, which contains 52 weeks. We model each week as one period. Each period starts from Monday and ends on Sunday. During weekends, there are no other business operations except transportation.

The complexity of the products in ABC has been taken into account. A set of standard products are identified and defined in the simulation model. Besides, specific product configurations defined by customers have also been taken into account in the model.

Three types of transportation modes are modelled: ocean, truck and air. Ocean mode is mainly used for the transportation between suppliers and manufacturers. Although it usually takes a few weeks, the freight cost is much lower than other transportation modes. Another important advantage of using ocean-shipping mode is that the corresponding carbon emission is significantly lower than other modes. Transportation by air is very expensive with extensive carbon emission. Air is only used for fulfilment of emergency orders. Truck mode is used to ship finished goods from manufacturers to DCs, and from DCs to retailers. Figure 3.3 shows the usage of different transportation modes between different supply chain facilities.

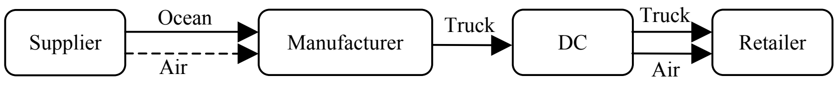


Fig. 3.3 Transportation modes

3.4.2 Scenario Analysis

One outstanding advantage of simulation is the capability to predict performance for TO-BE scenarios. After the baseline model was validated, different TO-BE scenarios were defined based on the baseline model. In each scenario, several aspects of the model were modified:

- inventory control policy;
- forecast accuracy;
- build to plan (BTP) vs. build to order (BTO);
- Procurement Policy;
- truck utilization.

In each scenario, a set of experiments were defined by setting different parameters. We ran each experiment, analysed the result, and finally came up with the scenario summary.

3.4.3 Inventory Control Policy

The DCs check their inventory position periodically and place purchase orders when inventory position falls below reorder point.

- On-hand inventory (I_h): quantity of unreserved products in the stock.
- In-transit inventory (I_t): quantity of purchased orders, which are being shipped.
- In-order inventory (I_o): quantity of purchase orders which have been received by suppliers but haven't been shipped out yet.
- Backlog orders (I_b): quantity of backlogged orders.

Inventory position (I_p) is defined using:

$$I_p = I_h + I_t + I_o - I_b. \tag{3.1}$$

When inventory position is lower than reorder point, a purchased order is placed. The requested quantity of the purchased order is determined by the inventory control policy.

Two basic inventory control policies were studied: (*s*, *S*) and (*R*, *Q*). In (*s*, *S*) policy, the reorder point is *s*, and the purchased quantity is given by:

$$Q_s = S - I_p. \tag{3.2}$$

In (*R*, *Q*) policy, the reorder point is *R*, and the purchased quantity is a constant number *Q*. The inventory control parameters can be calculated based on: desired service level, inbound lead time, outbound lead time, forecasted demand, and review period. In this study, we first calculate the inventory control parameters for (*s*, *S*) and (*R*, *Q*) policies respectively. Then we create several sub-scenarios by changing the parameters step by step. Finally the results are analysed to evaluate the impact on the inventory carrying cost and order fill rate. In this way, we can evaluate whether the model is sensitive to each inventory control parameter.

Figure 3.4 and Table 3.1 show the simulation results for different reorder points. According to the results shown below, we are able to determine the quantitative relationship between the reorder point and (1) inventory carrying cost; (2)

fill rate in the uncertain and dynamics environment. The decision makers would be able to decide the best combination of cost and fill rate.

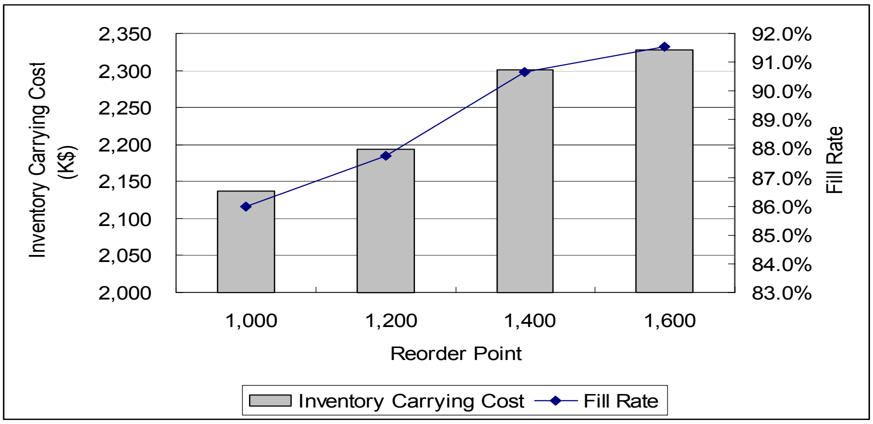


Fig. 3.4 Simulation results of inventory control policy scenarios

Table 3.1 Simulation results of inventory control policy scenarios

Reorder point	Inventory carrying cost (k$)	Fill rate (%)
1,000	2,137	86.0
1,200	2,193	87.8
1,400	2,301	90.7
1,600	2,327	91.6

3.4.4 Forecast Accuracy

The retailers are not part of the ABC Company. Orders are placed from the retailers to the DCs periodically. To maintain the service level and keep a low inventory level, the DCs need to forecast the demand of the retailers, so as to determine the inventory control policy.

In the simulation model, forecast accuracy (1-α) is an input parameter for the DCs. The actual retailer demand and the forecasted demand are both normally distributed, more specifically:

$$d_r \sim N(d_m, \delta^2) \quad \text{and} \quad d_f \sim N(d_r, \alpha^2), \tag{3.3}$$

where d_r is the actual demand quantity in the retailer orders, which is normally distributed with mean d_m and variance δ^2, and d_f is the forecasted order quantity in DCs.

DCs create purchase orders based on the forecasted demand. A buffer needs to be held in order to reach certain level of order fill rate. So the purchased quantity will be more than the forecasted quantity. With poor forecast accuracy, the order fill rate will also be poor. And DCs have to raise the buffer to improve the fill rate, so the inventory level will also be higher.

Four sub-scenarios are simulated with forecast accuracy 65%, 70%, 75% and 80% respectively. The results are shown in Figure 3.5 and Table 3.2. The inventory position is measured by product value. Similar to the analysis above, when forecast accuracy is improved, the average inventory position decreases while the fill rate increases.

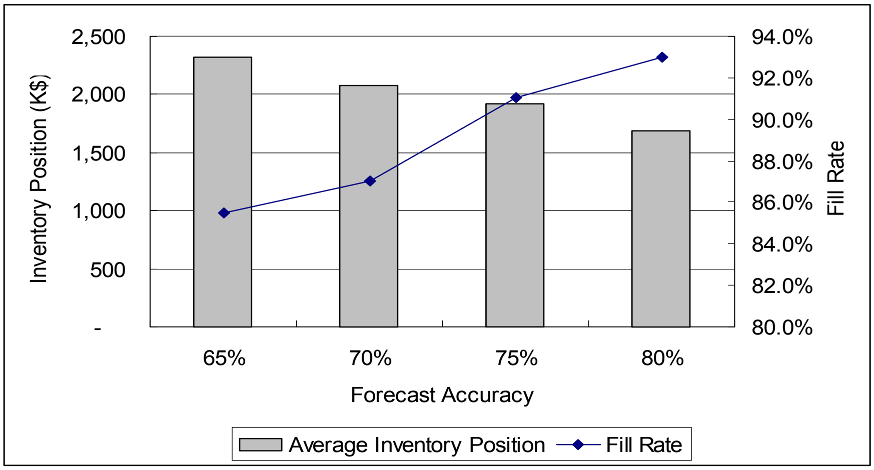


Fig. 3.5 Simulation results of forecast accuracy scenarios

Table 3.2 Simulation results of forecast accuracy scenarios

Forecast accuracy (%)	Average inventory position (k$)	Fill rate (%)
65	2,318	85.5
70	2,077	87.0
75	1,917	91.1
80	1,692	93.0

3.4.5 Build-to-plan vs. Build-to-order

The manufacturers can apply either BTP mode or BTO mode for the production process. In the BTP mode, manufacturers forecast DCs' demand and run an MRP (material requirements plan) periodically. The manufacturing process will be organized exactly based on the plan. In the BTO mode, instead of running MRP, manufacturers only produce finished goods when inventory level is lower than the predefined reorder point.

Theoretically, if the demand is relatively smooth, and the variances of inbound and outbound lead times are not significant, the production plan could be very accurate. However, the business reality is the opposite. A considerable investment in buffer inventory is necessary in order to keep a reasonable service level.

For simulating the BTP scenario, an MRP engine is embedded in the simulation model. The outputs of the engine are procurement plans, i.e. when to place purchase orders and how many components need to be purchased in each purchase order. The algorithm of the engine is:

1. Forecast the demand within the period.
2. Estimate the outbound lead time and calculate the shipping time.
3. Estimate the production time and calculate the release time (time to start producing finished goods).
4. Expand the bill-of-materials and calculate the required quantity for each component.
5. Estimate the inbound lead time for each component and calculate the time to place purchase orders.

In this scenario, we would like to compare the supply chain performance in BTP mode and BTO mode under different variances of demand and lead time. For this purpose, three variance profiles are designed: low, medium, and high. The medium profile is defined based on the AS-IS model. For the low profile, both demand variance and lead-time variance are decreased. For the high profile, the variances are increased. For each profile, we run simulations for both BTP mode and BTO mode respectively.

Figure 3.6 and Table 3.3 present the simulation results. According to the results, in a steady environment with low variance, both BTP and BTO perform very well. However, when the variances are higher, the fill rate for BTO mode is higher, and the cost for BTP mode is lower.

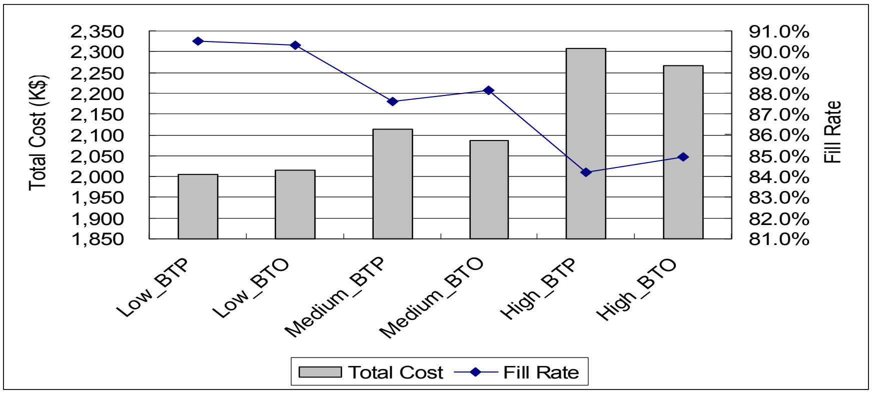


Fig. 3.6 Simulation results of BTP and BTO scenarios

Table 3.3 Simulation results of BTP and BTO scenarios

Variance	Mode	Total cost (k$)	Fill rate (%)
Low	BTP	2,005	90.5
Low	BTO	2,015	90.3
Medium	BTP	2,114	87.6
Medium	BTO	2,086	88.1
High	BTP	2,308	84.2
High	BTO	2,266	84.9

3.4.6 Procurement Policy

In a supply chain, each manufacturing node may have several upstream nodes. When placing orders, one or more nodes may be selected as order receivers. Single-sourcing policy or multiple-sourcing policy can be applied accordingly.

When the single-sourcing policy is applied, in the beginning of each period, each manufacturer will determine one supplier for the period. Purchase orders will only be sent to this supplier within the period. When the period is finished, another supplier could be selected for the next period.

When the multiple-sourcing policy is applied, each manufacturer will select a set of suppliers in the beginning of each period. A “sourcing proportion” is determined for each supplier. For a supplier, the sourcing proportion can be different for different manufacturers. The supplier set and the sourcing proportion will be changed when a new period starts. The sourcing proportion can be used in two different modes, “split” and “random”. In the split mode, purchase orders will be placed to all upstream nodes with positive sourcing proportions. The sourcing proportion determines the quantity of the purchase order. The order for the node with higher sourcing proportion will request more products. In the random mode, each time when a purchase order needs to be placed, an upstream node will be randomly selected. The probability is based on the sourcing proportion. The node with higher sourcing proportion will be more likely to be selected.

Three TO-BE scenarios are created, in which the procurement policies used are respectively single-sourcing policy, multiple-sourcing policy in split mode, and multiple-sourcing policy in random mode. Figure 3.7 and Table 3.4 display the simulation results. Comparing to multiple source policy, the cost of single-sourcing policy is higher, and the fill rate is lower. The differences between split mode and random mode are not very significant.

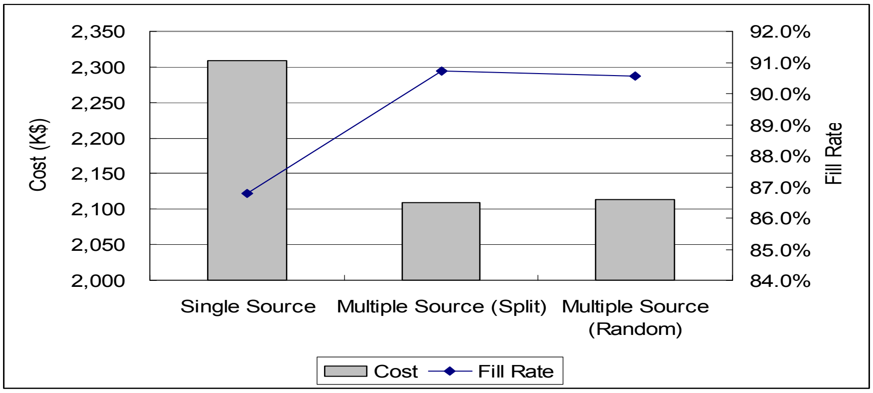


Fig. 3.7 Simulation results of procurement policy scenarios

Table 3.4 Simulation results of procurement policy scenarios

Procurement policy	Cost (k$)	Fill rate (%)
Single source	2,308	86.8
Multiple source (split)	2,109	90.7
Multiple source (random)	2,113	90.6

3.4.7 Truck Utilization

A DC usually delivers finished goods to retailers by trucks. Some trucks may be owned by the DCs. In some other cases, trucks could also be rented from external logistics companies. The delivery team needs to determine how many trucks are required. It's obvious that although it helps to improve customer service level with more trucks in place, the operation cost will be fairly high when the truck fleet utilization rate is low.

Figure 3.8 shows the operation process of the truck pool. Delivery requests are sorted in a priority queue. Actually this is a typical resource management system, and the concept can be expanded to other kinds of resources.

Truck utilization rate and average waiting time are key performance indicators of the truck pool. For each truck, the truck utilization is defined as $u = t_b / t_w$, where t_b is the total busy time, and t_w is the total working time. The average waiting time of delivery requests is the average time from entering the queue till the moment when assigned to a truck.

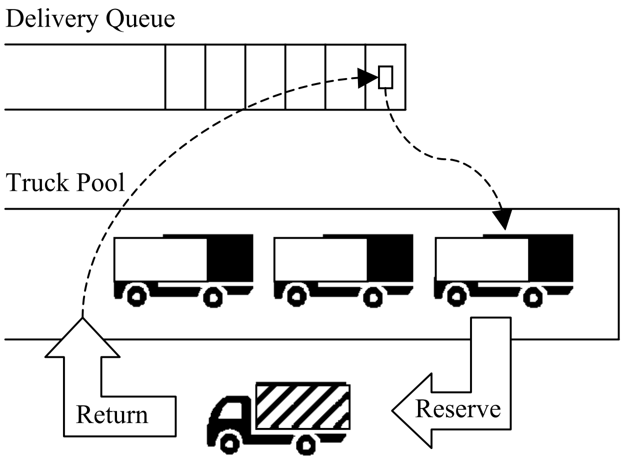
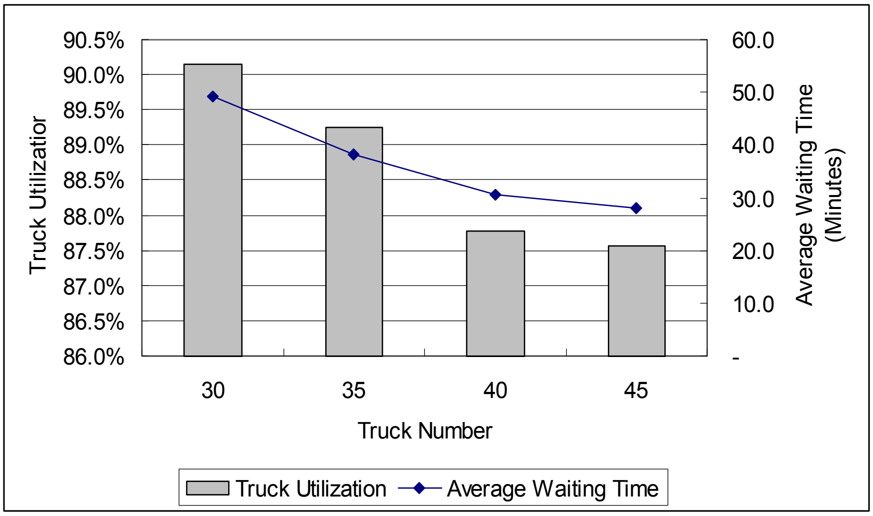


Fig. 3.8 Truck pool

Four TO-BE scenarios are defined using different number of trucks. Figure 3.9 and Table 3.5 show the simulation results. The results show that, when increasing the number of trucks, the waiting time decreases and the fill rate is improved. However the truck utilization decreases and the cost also increases.

Table 3.5 Simulation results of truck utilization scenarios

Number of trucks	Truck utilization (%)	Average waiting time (minutes)	Cost (k$)	Fill rate (%)
30	90.2	49.4	2,027	85.7
35	89.3	38.3	2,130	87.3
40	87.8	30.6	2,290	89.9
45	87.6	28.0	2,310	90.3



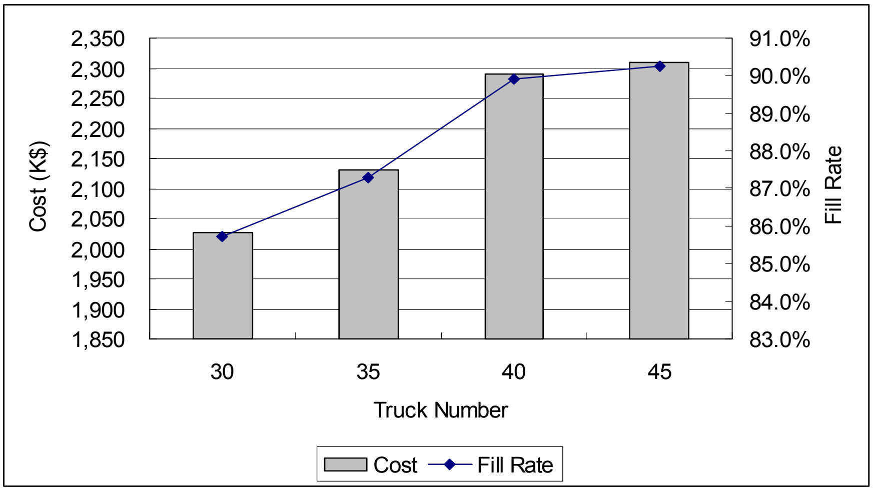


Fig. 3.9 Simulation results of truck utilization scenarios

3.5 Conclusions and Perspectives

Fierce competition and shorter product lifecycle force modern manufacturing enterprises to innovate and react to markets more responsively and intelligently, especially for those networked enterprises with global existence. Given the complexity, volatility and dynamics of the business arena, multi-agent-based simulation is a very practical technique for decision support. We introduced a multi-agent based simulation tool in this chapter, with a case study on a leading high-tech manufacturer. The project results show that networked enterprises can really get much insight from such quantitative analysis and would be able to identify lots of opportunities for cost saving and performance improvement.

Regarding future directions of this research, we plan to enhance the simulation library, especially the detailed models and policies for each agent. Furthermore, we'd like to apply such techniques for more clients and gain better understanding on how such technology would help decision-making in real industrial environments.

References

Bagchi S, Buckley SJ, Ettl M et al. (1998) Experience using the IBM supply chain simulator. In: Proceedings of the 1998 Winter Simulation Conference, pp. 1387–1394

Banks J, Buckley S, Jain S et al. (2002) Panel session: opportunities for simulation in supply chain management. In: Proceedings of the 2002 Winter Simulation Conference, pp. 1652–1658

Brun A, Cavalieri S, Macchi M et al. (2002) Distributed simulation for supply chain coordination. In: Proceedings of the 12th International Working Seminar on Production Economics, Igls, Austria

Chen F, Drezner Z, Ryan JK et al. (2000) Quantifying the bullwhip effect in a simple supply chain: The impact of forecasting lead times, and information. Mgmt Sci., 46(3):436–443

Cloutier L, Frayret JM, D'Amours S et al. (2001) A commitment-oriented framework for networked manufacturing coordination. Int. J. Comput. Integr. Manuf. 14:522–534

Dong J, Ding H, Ren C et al. (2006) IBM SmartSCOR–A scor based supply chain transformation platform. In: Proceedings of the 2006 Winter Simulation Conference, pp. 650–659

Fox MS, Barbuceanu M, Teigen R (2000) Agent-oriented supply chain management. Int. J. Flex. Manuf. Syst. 12(2/3):165–188

Gan BP, Low Y, Lim C et al. (2000) Parallel discrete-event simulation of a supply chain in semiconductor industry. In: Proceedings of HPC ASIA

Kaelbling LP, Littman ML, Moore AW (1996) Reinforcement learning: A survey. J. Artif. Intell. Res., 4:237–285

Kimbrough SO, Wu D, Zhong F (2002) Computers play the beer game: can artificial agents manage supply chains?. Decis. Support Syst., 33(3):323–333

Kleijnen JPC (2003) Supply chain simulation: A survey. Int. J. Simul. Process Mod., 103:1–20

Law AM, Kelton WD (2000) Simulation modeling and analysis, third edition, McGraw-Hill, Boston, MA

Lendermann P, Julka N, Gan BP et al. (2003) Distributed supply chain simulation as a decision-support tool for the semiconductor industry. Simulation 79(3):126–138

Mondal S, Tiwari MK (2003) Formulation of mobile agents for integration of supply chain using the KLAIM concept. Int. J. Prod. Res., 41(1):97–119

Nissen ME (2000) Agent-based supply chain disintermediation versus reintermediation: Economic and technological perspectives. Int. J. Intell. Syst. Account., Fin. Mgmt, 9:237–256

Pontrandolfo P, Gosavi A, Okogbaa OG et al. (2002) Global supply chain management: a reinforcement learning approach. Int. J. Prod. Res., 40(6):1299–1317

Qiu M, Ding H, Dong J et al. (2007) Impact of business service modes on distribution systems: a reinforcement learning approach. In: Proceedings of the International Conference on Services Computing, July

Repast (2008) http://repast.sourceforge.net/

Rubinstein RY, Kroese DP (2007) Simulation and the Monte Carlo method. Wiley-Interscience

Sadeh-Koniecpol N, Hildum D, Kjenstad D et al. (1999) MASCOT: An agent-based architecture for coordinated mixed-initiative supply chain planning and scheduling. In: Proceedings of the 3rd International Conference on Autonomous Agents (Agents '99), May

Scalable Simulation Framework (SSF) (2008) http://www.ssfnet.org/homePage.html

Shen W, Kremer R, Ulieru M et al. (2003) A collaborative agent-based infrastructure for internetenabled collaborative enterprise. Int. J. Prod. Res., 41(8):1621–1638

Sudra R, Taylor SJ, Janahan T (2000) Distributed supply chain simulation in GRIDS. In: Proceedings of the 2000 Winter Simulation Conference, pp.356–361

Sutton RL, Barto AG (1998) Reinforcement leaning–An introduction, MIT Press, Massachusetts

Swaminathan JM (1997) Modeling supply chain dynamics: a multi-agent approach. Decis. Sci., 29(3):607–632

Terzi, S, Cavalieri S (2004) Simulation in the supply chain context: A survey. Comput. Ind., 53: 3–16

Towill DR, Naim MM, Wikner J (1992) Industrial dynamics simulation models in the design of supply chains. Int. J. Physical Distrib. Logist. Manag., 22(5):3–13

Tzafestas S, Kapsiotis G (1994) Coordinated control of manufacturing/supply chains using multi-level techniques. Comput. Integr. Manuf. Syst., 7(3):206–212

Wang W, Dong J, Ding H et al. (2008) An introduction to IBM general business simulation environment. In: Proceedings of the 2008 Winter Simulation Conference, pp.2700–2707

Chapter 4
A Collaborative Decision-making Approach for Supply Chain Based on a Multi-agent System

Y. Ouzrout, A. Bouras, E.-H. Nfaoui[1] **and O. El Beqqali**[2]

Abstract To improve the supply chain's performance under demand uncertainty and exceptions, various levels of collaboration techniques based on information sharing were set up in real supply chains (vendor-managed inventory, continuous replenishment program, collaborative planning, forecasting and replenishment (CPFR),…). The main principle of these methods is that the retailers do not need to place orders because wholesalers use information centralization to decide when to replenish them. Although these techniques could be extended to a whole supply chain, current implementations only work between two business partners. With these techniques, companies electronically exchange a series of written comments and supporting data, which includes past sales trends, scheduled promotions, and forecasts. This allows participants to coordinate joint forecasting by focusing on differences in forecasts. But if the supply chain consists of autonomous enterprises, sharing information becomes a critical obstacle, since each independent actor is typically not willing to share with the other nodes its own strategic data (as inventory levels). That is why researchers proposed different methods and information systems to let the members of the supply chain collaborate without sharing all their confidential data and information. In this chapter we analyze some of the existing approaches and work and describe an agent-based distributed architecture for the decision-making process. The agents in this architecture use a set of negotiation protocols (such as firm heuristic, recursive heuristic, CPFR negotiation protocol) to collectively make decisions in a short time. The architecture has been validated on an industrial case study.

Keywords Multi-agent systems, supply chain management, decision-making, distributed simulation

---

[1] Y. Ouzrout, A. Bouras (✉), E.-H. Nfaoui
LIESP Laboratory – Université de Lyon – Université Lumière Lyon 2, 160 Bd de l'université 69676 Bron, France
e-mail: Abdelaziz.Bouras@univ-lyon2.fr

[2] O. El Beqqali
GRMS2I – FSDM/ Université Sidi Md Ben Abdellah, B.P 1796 Fès-Atlas, Morocco

4.1 Introduction

The economic and industrial communities worldwide are confronted with the increasing impact of competitive pressures resulting from the globalization of markets and supply chains for product fulfillment. More and more companies are being driven to pursue new forms of collaboration and partnership with their direct logistics counterparts. As a result, at a company level there is a progressive shift towards an external perspective with the design and implementation of new management strategies, which are generally named supply chain management (SCM). However, in order to conduct this concept in practice, several hurdles are still to be overcome (Samii, 2004; Hieber, 2002), mainly due to:

- The conflicts resulting from local objectives versus network strategies, because supply chain is a multi-decisional context, so companies must make decisions collectively.
- The difficulty in making decisions in a collaborative manner. It can be observed in several supply chain cases.
- The need for sharing sensitive information of participants in the supply chain. If the supply chain is composed of independent companies, sharing information becomes a critical obstacle, since each independent actor is typically not willing to share with the other nodes its own strategic data (production capacity, internal lead times, production costs, sales forecasts, etc.).
- The need for sharing information technology tools.

SCM has demanded attention and support from the industrial community. It consists in the coordination of production, inventory, location, and transportation among the participants in a supply chain to achieve the best mix of responsiveness and efficiency for the market being served (Hugos, 2003a). The optimal deployment of inventory is one of the main goals of SCM. Indeed, many collaborative processes (e.g., CPFR: collaborative planning, forecasting and replenishment (VICS Association, 2009), VMI: vendor-managed inventory (John Taras CPIM, 2009), continuous replenishment program and efficient consumer response (ECR, 2006)) and software systems (e.g., advanced planning systems (Simchi-Levi *et al.*, 2000), ERP: enterprise resource planning (Baglin *et al.*, 2001)) are used for management and control of inventory in order to reduce the total system inventory cost while maintaining the service levels that customers require.

Literature shows that the common objectives of the above practices are: elimination of surplus inventory, prevention of inventory shortage, minimization of safety stock, production and delivery of products in the right quantities and at the right time. However, for the distributor centers, it is difficult to achieve these goals, because of the challenges faced due to the frequent occurrence of unexpected rush orders from the wholesalers. These challenges will vary from one company to another and from one supply chain to another. In fact, the distributor cannot predict the date and the quantity of this type of order since it is a random one caused by multiple factors and depends closely on the branch of industry concerned. In addition, generally this type of order has a very short delivery date. In such emergency cases, the distributor may not be able to wait for the next planned delivery of products from the supplier. Therefore, generally the order can be cancelled or can cause an inventory shortage if the ordered quantity is large. This will have a bad impact on the quality of the offered service within the satisfaction of the final customer policy. Moreover, some suppliers allocate additional human resources and logistics for delivering the unexpected rush orders from their distributors. The disadvantage of this solution is that the costs suggested are generally very high.

In multi-echelon network, which is a common distribution model for many distributors and manufacturers, the distributors can deliver the unexpected rush orders. The echelon inventory includes the sum of local stock and the stock of all the forward distribution centers (Siala *et al.*, 2006; Lee, 2003). However, multi-echelon inventory management is more consistent with the centralized decisions. It requires that all locations must be submitted to the relevant control of a single company. In addition, it requires a high degree of information sharing between the various actors of the supply chain, but if the supply chain consists of independent companies, information sharing becomes a critical obstacle, since each independent actor is typically not willing to share with the other nodes its own strategic data (as inventory levels). Also, it monitors its inventory levels (by using autonomous action and policies) and places the orders to its suppliers in order to optimize its own objective (Siala *et al.*, 2006).

This chapter focuses on unexpected swings in demand and on unexpected exceptions (problem of production, problem of transportation, etc.), which are important coordination and communication issues in SCM (Giannoccaro *et al.*, 2003; Zhao *et al.*, 2002; Reutterer and Kotzab, 2000). Both events can engender the presence of unexpected rush orders in a node of the supply chain; in particular, at the wholesalers and the distribution center levels. In this context, we propose a collaborative process which presents an effective solution (to the distributors) for better management of the unexpected rush orders for which the quantity of product cannot be delivered partially or completely from the available inventory. This process includes the distributors of the same or equivalent products and their wholesalers. The participants in the process can be competitors. To implement the process, we apply an agent-based distributed architecture in order to guarantee the autonomy and the strategic data confidentiality of all participants. Agent technology provides to the distributed environment a great promise of effective communication (Swaminathan *et al.*, 1998). An agent is a program that performs a specific task intelligently without any human supervision and can cooperatively communicate with other agents. Therefore, agent technology is suitable to solve communication concerns for a distributed environment. Recent research also shows that the multi-agent approach plays a significant role in SCM (Kimbrough *et al.*, 2002; Wu, 2001; Fu *et al.*, 2000; Swaminathan *et al.*, 1998).

The remainder of the chapter is structured as follows. Sect. 4.2 introduces the role of distributed simulation in the supply chain context. Sect. 4.3 shows generic model fixing and defining the supply chain elements to be considered and the modeling methodology. Sect. 4.4 presents the distributed decision-making architecture based on a multi-agent system. Sect. 4.5 illustrates the implementation of the system on an industrial case study. Finally, Sect. 4.6 concludes the chapter and gives an overview of the research perspectives.

4.2 Distributed Simulation and Supply Chain Management

Many software vendors (e.g., IBM (Bagchi *et al.*, 1998)), universities and companies (Telle *et al.*, 2003; Banks *et al.*, 2002) have traditionally used a local simulation approach in the supply chain context. Only in recent years, more and more companies in supply chains have adopted distributed collaborative simulation (Brun *et al.*, 2002), because it provides a connection between supply chain nodes that are geographically distributed throughout the globe, guaranteeing that each single simulation model is really linked to its respective industrial site. Moreover, as the companies in any supply chain make decisions individually as well as collectively regarding their actions in production, inventory, location, transportation and information, the use of distributed simulation can help in preservation of the local autonomies and privacy of logistics data. In some cases, the execution of a distributed model allows one to reduce the time spent for simulation, since separated models run faster than a single complex model (Fujimoto, 1999).

Despite the great use of simulation in supply chains and SCM, there are many additional opportunities for application of the methodology (Banks *et al.*, 2002). However, many of these opportunities require that challenges be overcome (see Introduction). Aims of the agent-based distributed simulation presented in this chapter are:

1. to convince decision-makers to adopt an SCM process and to choose the most appropriate management strategies and practices for a given supply chain;
2. to make decisions collectively in a short time; mainly, in the case of operational planning (short term, for example: rush order) (Pinedo and Chao, 1999) or in a situation where the supply chain partners negotiate a delivery date modification due to a disturbance (for example: problem of production, problem of transport, etc.), where the decision system has to make its choice within a short time, while evaluating the consequences regarding various scenarios in a distributed manner.

4.2.1 Decision-making and Multi-agent Approaches

In order to have a flexible and proactive model, we have chosen the multi-agent approach to develop our architecture. Cloutier *et al*., (2001) and Parunak (1996) showed the main benefit of using this approach in the field of the supply chain. Thus, the complete architecture of simulation is made on a set of agents modeling the supply chain participants. These collaborative agents communicate between them and negotiate using protocols. They seek the accurate and timely data that hold the promise of better coordination and better decision-making in the information systems of the supply chain participants (such as an ERP system); this means every time the simulation starts, the model must be initialized with the current states of the supply chain participants.

In the literature, various researches have been proposed to compensate for the uncertainty that exists in a supply chain. Cohen and Lee (1988) have developed a planning model to optimize material supply, production and distribution processes. Arntzen *et al.* (1995) have proposed a resource allocation and planning model for global production and distribution networks.

Kimbrough *et al.* (2002), McBurney *et al.* (2002) and Chen *et al.* (2000a, 2000b) focused on demand forecasting. Most of these studies suppose that companies in the supply chain share the information and coordinate the orders. But if the supply chain consists of autonomous companies, sharing information becomes a critical obstacle, since each independent actor is typically not willing to share with the other nodes its own strategic data (as inventory levels) (Terzi and Cavalieri, 2004). An example is the case of several competitor wholesalers (located in the same or different geographical areas) which source of the same distributor.

4.2.2 Multi-agent Systems Simulation

Various projects applied the multi-agent system paradigm to solve different problems in supply chains (inventory planning, demand and sales planning, distribution and transportation planning, etc.). DragonChain is implemented by Kimbrough *et al.* (2002) at the University of Pennsylvania (Philadelphia, PA, USA) to simulate SCM, and more particularly to reduce bullwhip effect. For that, they base their simulation on two versions of the Beer Game, the MIT Beer Game (i.e. the original game) and the Columbia Beer Game, and they use agents that look for the best ordering scheme with genetic algorithms.

Maturana *et al.* (1999) developed a hybrid-agent-based mediator-centric architecture, called MetaMorph, to integrate partners, suppliers and customers dynamically with the main company through their respective mediators within a supply chain network via the Internet and intranets. In MetaMorph, agents can be used for representing manufacturing resources (machines, tools, etc.) and parts, encapsulating existing software systems and functioning as system or subsystem coordinators/mediators.

A multi-agent approach to model supply chain dynamics is proposed by Swaminathan *et al.* (1998). In their approach, a supply chain library of software components, such as retailers, manufacturers, inventory policy, and so on, has been developed to build customized supply chain models from the library.

Sadeh *et al.* (2001) developed an agent-based architecture for a dynamic supply chain called MASCOT. MASCOT is a reconfigurable, multilevel, agent-based architecture for a coordinated supply chain. Agents in MASCOT serve as wrappers for planning and scheduling modules. Petersen *et al.* (2001) have proposed a multi-agent architecture, called AGORA, for modeling and supporting cooperative work among distributed entities in virtual enterprises.

Moreover, Nfaoui *et al.* (2006) proposed an agent-based distributed architecture for simulation in decision-making processes within the supply chain context. Agents in this architecture use a set of negotiation protocols (such as firm heuristic negotiation, recursive heuristic negotiation, CPFR negotiation protocol) to make decisions collectively in a short time.

Chehbi *et al.* (2003) proposed a multi-agent supply chain architecture to optimize distributed decision-making. Furthermore, Moyaux *et al.* (2004) developed an agent simulation model for the Québec forest supply chain.

4.3 The Supply Chain Modeling

Two parameters are important in the process of the modeling of a supply chain, the perimeter and the structure. The first delimits the supply chain in terms of number of actors (companies), and the second defines the customer/supplier relationships. If the two aforementioned parameters are not described, it is difficult to define the modeling boundaries (in terms of levels). Indeed, a supply chain can use several tens, even hundreds of nodes geographically distributed throughout the globe.

1. Is it really necessary to take into account all the actors?
2. Moreover, a company can belong to several supply chains. Then, what are the levels of customers and suppliers that should be covered by the proposed model?

Figure 4.1 presents a simple example of a global distribution supply chain.

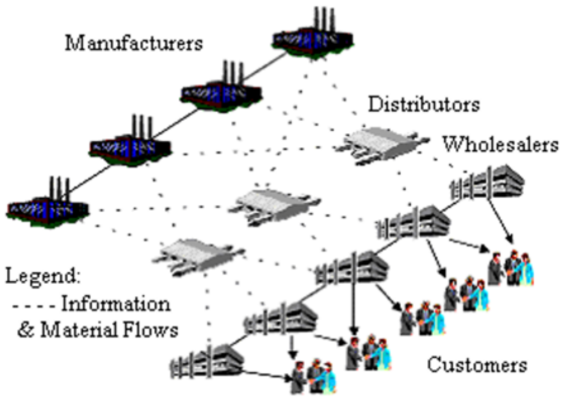


Fig. 4.1 Overview of a global distribution supply chain, information and material flows

4.3.1 Supply Chain Modeling Methodology

To answer these two questions, it is necessary to identify the product for which the supply chain is defined. Indeed, we define a supply chain for a product or a family of products. It is composed of all the companies involved in the design, production, transportation, and delivery of a product to market. Having a clear knowledge of the product, we better specify the central company of the supply chain, i.e. the one that assembles the finished product. Next, we propose to follow the steps below, they define a generic model for supply chain, which delimits the boundaries of modeling and defines the customers/suppliers relationships:

1. Identify the finished product for which the supply chain is defined. This automatically defines the core company (central company) of the supply chain.
2. Identify the bill of materials of the finished product.
3. Exclude from this bill of materials the raw materials not requiring a partnership or collaboration.
4. Afterwards, identify the remaining raw materials suppliers (distributors or factories having the activity of production). The same raw material can be bought from one or more different suppliers. In this last case, the percentage of the orders to place to each one of them should be determined.
5. For each supplier, steps 2, 3 and 4 have to be renewed by considering, this time, the raw material as a finished product, and so on up to the upstream supplier.
6. Determine the type of the orders (stationary, random, etc.) of the customers (other than central company) of the various suppliers identified in step 4.
7. Redo step 6 for the suppliers of the suppliers except for the last suppliers (upstream suppliers).
8. Identify the list of customers of central company.

9. In this list, identify the potential customers acting on the supply chain of the product (customers requiring collaboration or a partnership) and those, which do not require the collaboration. Then, determine the type of the orders for the latter (random, stationary, etc.), as well as the percentage of the orders, which each potential customer places to central company.
10. For each potential customer of central company, remake steps 8 and 9 except for the final customers (for example, consumers).
11. Allot an agent-based model to all the identified actors (central company, customers and suppliers).

To represent the three main functions of the company (source, make and deliver) and to consider the control processes in the supply chain and its environment, each actor is modeled by seven agents. These agents are: AgentPRC, which plays the part of the processes related to the customer, AgentDis, which manages the "distribution storage", AgentPro, which plays the role of the "make" process, AgentApp for the "source" process, AgentAch, which plays the role of the "purchase" process, AgentSCM for the "management" of the supply chain and AgentPer, which handles the "disturbances". This last agent makes it possible for the model to be open and extensible in order to consider a large variety of disturbances so as to cover various types of supply chain (world size, national size, branch of industry, etc.). In the case of an actor of the distributor type, the agents AgentPro and AgentApp do not exist. Figure 4.2 shows the implementation level of the AUML (Agent Unified Modeling Language) class diagram for the agent AgentSCM.

4.3.2 The Safety Inventory Case Study

To illustrate this methodology we use an industrial example based on the unexpected rush orders and the safety inventory management. Safety inventory is necessary to compensate the uncertainty that exists in a supply chain. Retailers and distributors do not want to run out of inventory in the face of unexpected customer demand or unexpected delay in receiving replenishment orders so they keep safety stock on hand. A thumb rule for deciding the level of safety stock may be, "the higher the level of uncertainty, the higher the level of safety stock required".

Safety inventory for an item can be defined as the amount of inventory on hand for an item when the next replenishment economic order quantity lot arrives. This means that the safety stock is an inventory that does not turn over. Basically, it becomes a fixed asset and it drives up the cost of carrying inventory. Companies need to find a balance between their desire to carry a wide range of products and offer high availability on all of them and their conflicting desire to keep the cost of inventory as low as possible (Hugos, 2003b). That balance is reflected quite literally in the amount of safety stock that a company carries.

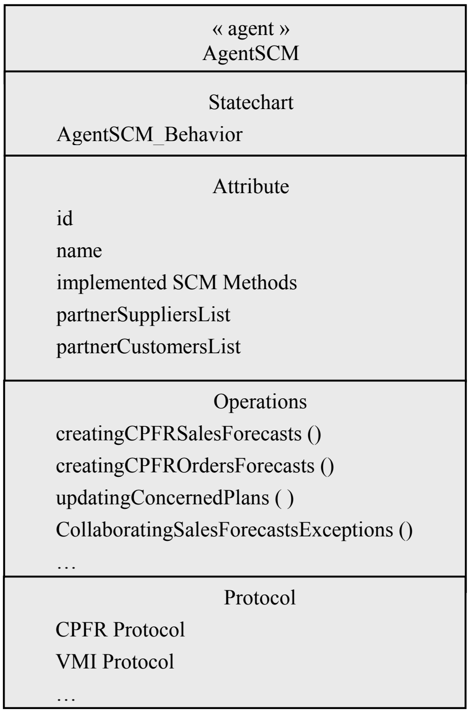


Fig. 4.2 Example of a class diagram (implementation level) for the "AgentSCM"

In practice, the safety stock is not enough to cover all types of unexpected swings in demand and the unexpected exceptions. As an example, the case of a customer (retailer, wholesalers, etc) who contacts his supplier (distributor) and asks for a product quantity as an immediate request, and the supplier discovers that his safety stock is lower than the ordered quantity at that moment. If this happens after the item has been logged in as a confirmed order, will the supplier be able to respond within a suitable timeframe to the customer?

We propose hereafter an agent-based distributed architecture to solve this problem of unexpected rush orders. The unexpected rush order is an object, which is characterized by two attributes, the ordered quantity and the short delivery time. Multiple causes of unexpected rush order exists, they closely depend on the branch of supply chain sector. For distributors, the main interest got from such unexpected rush order delivery depends on the customer's profile and other additional criteria such as:

- It allows interesting benefits.
- The distributor does not look for benefits, but he only seeks to make the customer faithful. It is the case for a wholesaler strategic customer for example.
- It helps a wholesaler to get rid of his surplus inventory.

- It helps to increase the number of customers. It is the case of a wholesaler who is a customer of another distributor or a regular customer who is in hurry, who will appreciate the offered service and may become a new customer.

4.4 The Multi-agent Architecture

4.4.1 The Modeling Principle

A distribution system needs to include one or more customers who are defined by having a demand on a given product. In addition, the system needs to include one or more sources, which are defined by producing or containing the product demanded by the customer(s). Finally, the system needs a connection between the source(s) and customer(s), which can accommodate a flow of the product from the source(s) to the customer(s) in order to obtain fulfillment of the demand.

Let us assume that a wholesaler or a particular customer had placed a unexpected rush order which is characterized by two attributes respectively:

- the ordered quantity (*OQ*) which cannot be delivered partially or completely from the distributor's available inventory; and
- the delivery time (*DT*).

Two cases are then possible:

1. The distributor does not have any part of the *OQ*.
2. It has a part of the ordered quantity and must complete the rest.

In both situations:

$$OQ = DisQ + RQ, \tag{4.1}$$

where *DisQ* corresponds to the available quantity which can be delivered by the distributor with possible situations: $DisQ = 0$ (a) or $DisQ < OQ$ (b), and *RQ* corresponds to the required quantity, which must be looked for. The problem can be then summarized as follows: find the *RQ* in order to deliver the unexpected rush order while respecting the *DT*.

One of the practices of the distributors consists of seeking the required quantity from another wholesaler or distributor. In general, the distributor will be limited to some close and faithful wholesalers or to distributors of the same company. This shows that the chosen solution (if it is found) cannot be the best. Moreover, it takes enough time since the negotiation is carried out generally by phone. Within the context of SCM, we propose to extend this practice by involving (into collaboration) several wholesalers, the same products distributors and equivalent products distributors. So, the distributor will be supplied from three different actors:

- 1st type (distributor and its wholesalers):
  - wholesalers belonging to the same area as the customer;
  - wholesalers belonging to different areas from the customer; and
  - both cases above.
- 2nd type (distributor/same products distributors):
  - the same products distributors and/or their wholesalers.
- 3rd type: (distributor/equivalent products distributors)
  - the equivalent product distributors and/or their wholesalers.
- 4th type (hybrid solution):
  - hybrid solution of the two or more preceding cases.

In practice, to implement this process in the industrial cases, and in order to satisfy the distributors' needs, three conditions must be checked:

- Quick and automatic solution: the manager contacts the collaborative participants only to confirm the solution.
- Autonomy and data confidentiality of each participant must be guaranteed since they can be independent.
- Transportation costs should be minimized, which will be added to the basic high purchase price since the products will not be delivered directly by the distributor.

As shown in the background review (see Sect. 4.2), the agent technology is more suitable for this type of problem. In this respect, we propose in the section below an agent-based distributed architecture, which implements these processes.

4.4.2 Architecture

The proposed system does not substitute for the existing tools and the SCM strategies and practices, but it can be used as a complement which improves them in the case of the presence of a unexpected rush order. The system architecture is shown in Figure 4.3.

The main purpose of this system is to coordinate all collaborative participants in order to find the *OQ* and minimize total cost of transportation when a unexpected rush order is presented. The complete architecture includes two types of agents: control agents *WhoAgent* for wholesalers and the unexpected order facilitator agents *DisAgent* for distributors. Each distributor is modeled by a $DisAgent_i$ (where $i \in [1, nd]$ and nd is the number of collaborative distributors) and each wholesaler is modeled by a $(WhoAgent_i)_j$ (where $j \in [1, nw_i]$ and nw_i is the total number of collaborative wholesalers of the $distributor_{(i)}$). These coordinated agents have the ability to specify both static and dynamic characteristics of various supply chain entities (Lee *et al.*, 1997); in particular, the level of distributors and their wholesalers.

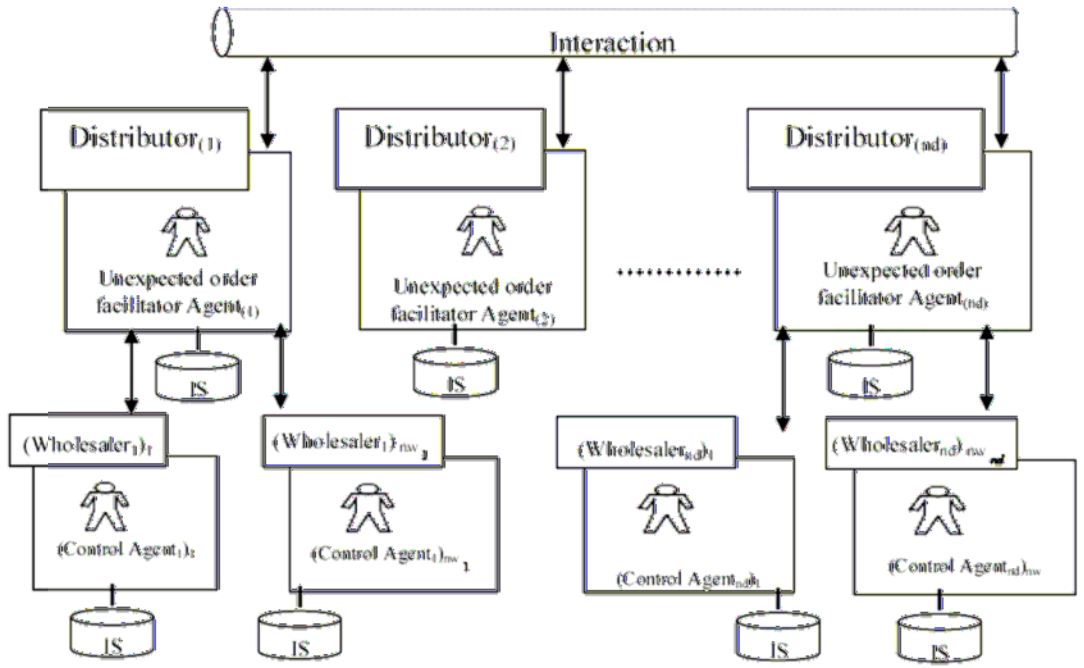


Fig. 4.3 Multi-agent system architecture.
nd : number of collaborative distributors;
nw_1: number of collaborative wholesalers of Distributor $_{(1)}$;
nw_{nd}: number of collaborative wholesalers of Distributor $_{(nd)}$;
IS: information system (such as an ERP system)

The control agent plays a liaison role between a supply chain manager and the system. It collects strategies from managers, seeks the accurate data and aims at building a rule-base for better coordination and better decision-making process. When an unexpected order facilitator agent $DisAgent_i$ asks a control agent $(WhoAgent_i)_j$ (where $j \in [1, nw_i]$ and nw_i is the total number of collaborative wholesalers of the distributor$_{(i)}$) for the possible quantity of product that it can deliver, control agent $(WhoAgent_i)_j$ sends information about the possible quantity and costs, and transportation costs. The agent system is autonomous because it allows any manager to change the strategies of that node. Also, it allows overcoming the hurdle, which consists of the need for sharing sensitive information of participants in the supply chain, because the agents do not exchange information about inventory levels and strategies.

In real-world coordination, sharing information truthfully is problematic since intra-organizational trust cannot be easily developed. Next, if the collaborative participants (wholesalers and their distributors) are independent and operate within the same sector (same or equivalent products), information sharing becomes a critical obstacle, since each independent wholesaler/distributor is typically not willing to share with the other nodes its own strategic data (inventory level for example). In the real global distribution cases, and especially in the same distribution sector, it is easy to implement the interaction between a distributor and its wholesalers. However, it is difficult, even impossible, to implement it between a distributor and the wholesalers of another distributor. Indeed, the participants are confronted with the increasing impact of competitive pressures. Also, each distributor tries to increase the number of its customers. To represent the real global distribution cases, each agent $DisAgent_i$ can interact with other agents $\{DisAgent_k / k \in [1, nd], k \neq i\}$ and control agents $\{(WhoAgent_i)_j / j \in [1, nw_i]\}$, but a control agent can interact only with its distributor agents.

The unexpected order facilitator agent $DisAgent_i$, which communicates with control agents and other unexpected order facilitator agents, plays the same role of control agent. In addition, it provides the best solution for solving the unexpected rush order, while minimizing the total transportation costs. When an unexpected rush order (characterized by OQ and DT) is presented at the distributor$_{(i)}$, the manager asks the unexpected order facilitator agent $DisAgent_i$ to search the best solutions. At this moment, the agent carries out the algorithm "search_OQ" for solving the problem, which can be summarized as follows:

1. Search the RQ. It can be delivered completely by only one wholesaler or gathered from several wholesalers (first type).
2. Find and classify series of possible paths in an ascending order depending on transportation cost (which depends closely on the distance) to transport OQ while respecting DT.
3. Propose quantity Q_j to be supplied by each participant belonging to a path.

Figure 4.4 shows the UML (Unified Modeling Language) sequence diagram (Bauer and Odell, 2005) that expresses the exchange of messages through the interaction protocol between the unexpected order facilitator agent $DisAgent_i$ and the control agents $\{(WhoAgent_i)_j / j \in [1, nw_i]\}$.

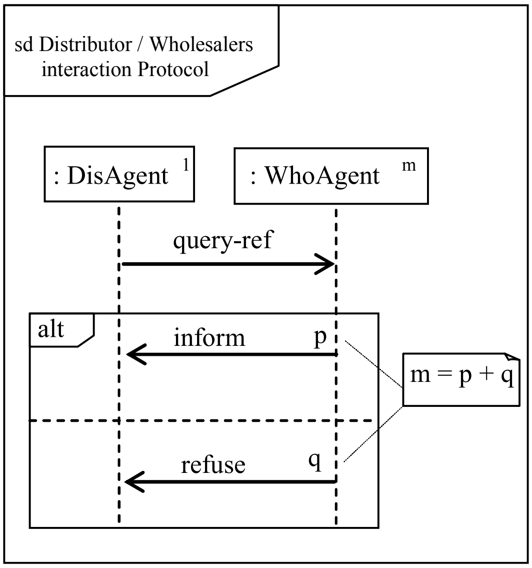


Fig. 4.4 Distributor/wholesalers interaction protocol

If no solutions are found or the manager is not satisfied with the proposed solutions, the agent can interact with the unexpected order facilitator agents $\{DisAgent_k / k \in [1, nd], k \neq i\}$ of other distributors in order to find the best solution. In this case, the agent $DisAgent_i$ sends a message of performative *QUERY-REF* to all the collaborative unexpected order facilitator agents to find the *RQ*. Each agent applies his decision-making process in order to deliver the *RQ*. They can interact if necessary with the control agents to get the best solution. At the end, the unexpected order facilitator agent $DisAgent_i$ sorts all the received solutions. Figure 4.5 shows the UML sequence diagram that expresses the exchange of messages through the interaction protocol between the unexpected order facilitator agents of several distributors.

4.4.3 Negotiation Protocols

The negotiation is the mechanism by which the agents can establish a common agreement. In the case of intelligent agents and of the multi-agent systems, the negotiation is a basic component of the interaction because the agents are autonomous (Jenning *et al.*, 2001); there is no solution imposed in advance and the agents must find solutions dynamically, while solving the problems. To model the negotiation between the agents composing our system, we consider the following aspects:

- *The negotiation object*: an abstract object which includes the attributes that the agents want to negotiate. In our architecture, several objects are prone to be negotiated according to the situation. We find among others, the Order and its attributes (quantity and delivery date), the contract of continuous delivery and its attributes (quantities and plan of delivery), the forecasts and their attributes (quantities, dates and exceptions) and the acceptable scenario in the case of dysfunctions.
- *The decision-making process*: this is the model that the agent uses to make the decisions during the negotiation. The most important part of making decisions is the negotiation strategy which allows the agent to choose the most appropriate communicative intention (also called "performative") at a certain time. The performative can be ACCEPT_PROPOSAL, REQUEST, INFORM, PROPOSE, etc.

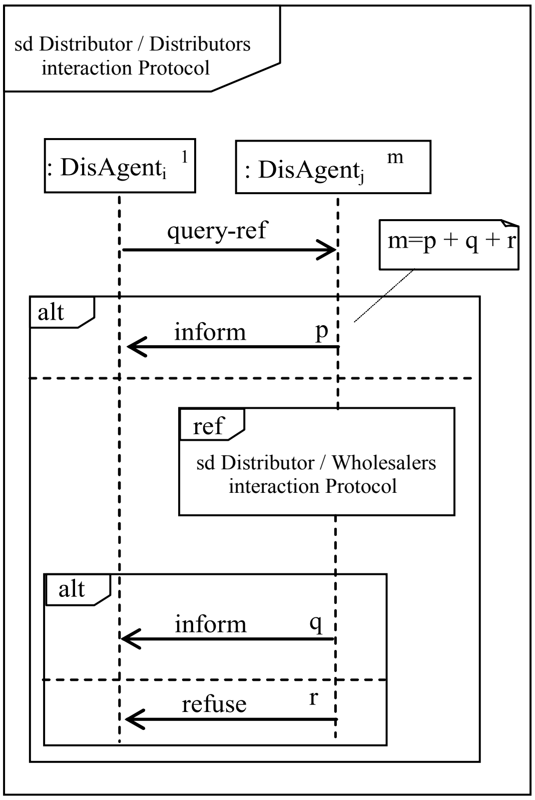

sd Distributor / Distributors interaction Protocol
: DisAgent$_i$ 1
: DisAgent$_j$ m
query-ref
m=p + q + r
alt
inform p
ref
sd Distributor / Wholesalers interaction Protocol
alt
inform q
refuse r

Fig. 4.5 Distributor/distributors interaction protocol

- *The communication language*: the language used by the agents to exchange their knowledge and information during the negotiation. We use the FIPA-ACL language (FIPA, 2002) in our application.
- *The negotiation protocol*: the set of elements that governs the negotiation such as the possible participants in the negotiation, the legal proposals that the participants can make, the states of the negotiation, and finally, a rule to determine when the negotiation should be stopped in case of agreement (or when it is necessary to stop the negotiation process because no agreement could be reached).

In the SCM process, the agents are cooperative, having the same goal (aggregation of the local objectives). They share and solve problems together. For this reason, the agents must provide useful reactions to the proposals that they receive. These reactions can take the form of a counter-proposal (refused or modified proposal). A counter-proposal is an alternative proposal generated in response to a proposal. From such reactions, the agent must be able to generate a proposal, which is probably ready to lead to an agreement. Consequently, the agents of our system must use protocols respecting the criteria which have been stated above and that mainly depend on three parameters:

1. The branch of supply chain sector (textile and clothing sector, consumer goods sector, etc.).
2. SCM strategies and practices used for the companies' cooperation and coordination.
3. Objects to be negotiated: rush order, ordinary order, sales forecasts, orders forecasts; modification of delivery plans in case of trouble, etc.

Heuristic Negotiation

Figure 4.6 shows, as an example, the heuristic negotiation (Florea, 2002). In this protocol several proposals and counter-proposals can be exchanged in various steps. Agent "A", with proposal "pA", is the initiator of the negotiation, whereas the agent "B" (participant) can reply with the answers "p1B", "p2B" and "p3B" (to modify the request). The number of the counter-proposals is limited. Once this limit is reached, the agents arrive to a rejection. We propose to recapitulate the heuristic negotiation protocol using UML sequence diagram (Figure 4.6).

4.5 Industrial Case Study

A case study of the proposed agent-based distributed architecture via conducted at a leading distribution company in North Africa. The aforementioned company operates in the sectors of toilets and showers (washbasin, baths, etc.), taps, tiling, erates in the sectors of toilets and showers (washbasin, baths, etc.), taps, tiling, plumbing and pieces of furniture. It offers about 800 economical and high-end items at both its wholesale and retail market locations, which are geographically distributed throughout the region. The products are sourced from two national suppliers (located in Morocco) and two main foreign suppliers (located in Spain and Turkey). The distributor mainly manages three types of independent-demand inventory systems. The periodic review system (sometimes called a fixed interval reorder system or periodic reorder system) is used to manage the inventory of the high-end products sourced from the foreign suppliers; a new order is always placed at the end of each review; the time between orders is fixed at 6 months for high-end items of plumbing, and 1 month for high-end items of tiling. The reorder point system (or fixed ordering quantity system), which tracks the remaining inventory of an item (sourced from national suppliers) each time a withdrawal, is made to determine whether it is time to reorder. Demands are estimated from forecasts and/or customer orders. A sourcing-to-order system is used for some items depending closely on taste and latest fashion (like products of tiling, etc.).

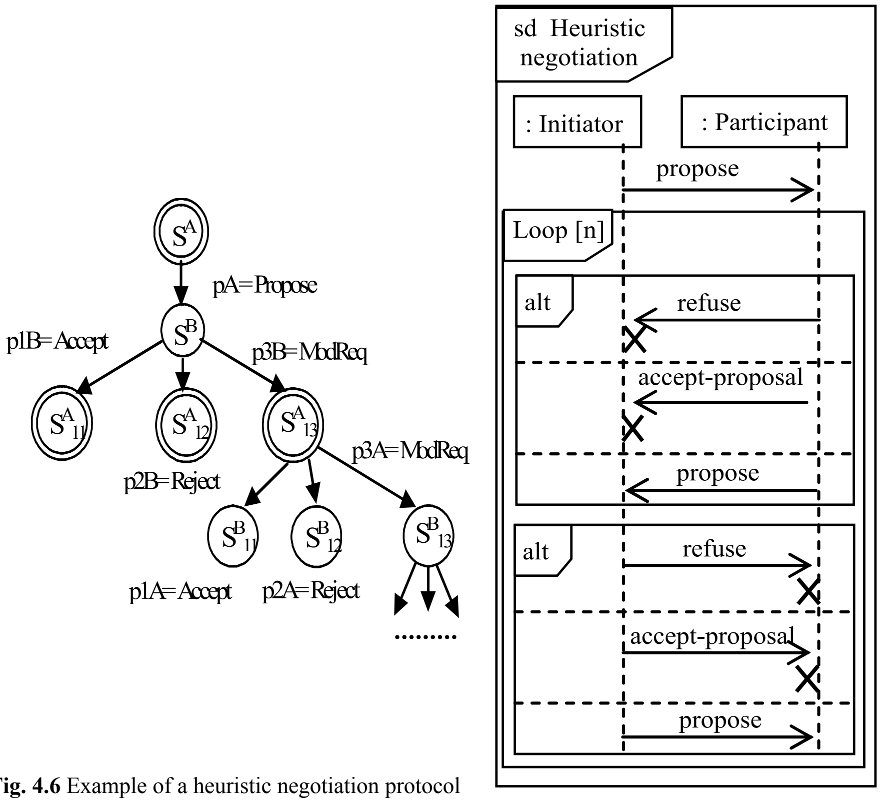


Fig. 4.6 Example of a heuristic negotiation protocol

The distributor has noticed that, in the whole, there are 6 to 10 unexpected rush orders that could take place each month. Actually, it is a big challenge because the aforementioned orders do not concern the same product. At the average level, 60% of such orders are canceled, 20% are made by other distributors, and the delivery time of the other 20% can be modified after a tough negotiation with the customer.

At the beginning of the year (the beginning of the low sales season), a real-estate operator (who already promised the delivery of the apartments to the customers at a precise time) must have closed the building site (located in Fez city in the central part of Morocco) during one week in order to have the housing license agreement as soon as a high-end item required in the construction had been in shortage. The latter is due to the inventory shortage of this item at the distributor. Because of a problem in the supplying process mainly due to one of the foreign suppliers, the next planned delivery of this item has been backlogged. In addition, the required high-end item cannot be supplied from other distributor because the aforementioned distributor is the only exclusive actor of such product. This order placed during the first week of the month was regarded as an unexpected firm rush order (delivery time: 9 hours; and it can be neither delayed nor canceled).

Adopted Solution

In this emergency case, the distributor was compelled to place an order of an equivalent item (mark, color…) at another competitor distributor located in Casablanca city (about 289 km from Fez city). Even if the costs were high, the distributor was interested in rendering the real-estate operator faithfully.

Solutions Provided by the Proposed System

In order to show the effectiveness and usefulness of the suggested architecture, we have made use (as a tangible demonstration) of the aforementioned item inventory-level history of 37 wholesalers distributed throughout the country. They are identified by scm1, scm2… scm37.

Thirteen existing solutions (paths and quantities) have been proposed by the unexpected order facilitator agent, according to the value of the number of wholesalers involved in each path. This number is chosen by the manager; $Q_{\min}$ is the minimal quantity of product which can be transported in a segment connecting a wholesaler and the next wholesaler in the same path.

Results Comments

- We notice that even if the required quantity cannot be delivered completely by only one wholesaler, it can gathered from both wholesalers scm26 (Ourzazate) and scm23 (Marrakech). This solution is not appropriate since the distance of the path is 687 km. Indeed, the costs of transportation will be high.

- The distributor has 12 choices concerning the paths including three wholesalers. The time of the first three paths are almost equal to half of the required delivery time. The distances (323 km, 343 km) of the first two paths (Tetouan–Chefchaouen–Meknes–Fez; Tanger–Tetouan–Chefchaouen–Fez) are about 289 km (distance between the competitor distributor and the customer). Certainly, these two solutions are better than the adopted solution, at least for two reasons:

  1. The costs of transportation will be almost the same.
  2. The purchase prices offered by the collaborative wholesalers (involved in the same supply chain) are lower than the offered prices from the competitor distributor.

We deduce that the distributor had three solutions better than the adopted one. We can conclude that as the number of actors constituting a path and the required quantity become large, the important more the number of choices. In practice, several emergency situations exist and depend closely on the branch of supply chain sector.

4.6 Conclusion and Perspectives

This chapter was dedicated to an agent-based distributed architecture for collaborative decision-making processes within the global distribution supply chain. An efficient process to manage the unexpected rush orders, using the properties of multi-agent, has been presented. A generic model, allowing a flexible modeling of the supply chain is used. This model makes it possible to test a set of negotiation protocols that have been previously proposed and modeled using UML. The used scenario shows that UML diagrams offer effective solutions to specify and model real-world agent-based applications. At this stage, the architecture was validated and tested on a leading distribution company. The results are very promising and the proposed system could be connected to the existing supply chain tools.

An enhancing stage could consider the size of the transported batches, i.e. to represent the price of transportation per unit of product as being lower when the transported quantity is large. In this case, it is necessary to seek the quantities, which optimize the costs of transportation (using for example genetic algorithms). Then, the presented architecture could be enhanced to be used in a global manufacturing supply chain (manufacturers, suppliers, etc.).

References

Arntzen BC, Brown GG, Harrison TP et al. (1995) Global supply chain management at digital equipment corporation. Interfaces 25(1):69–93

Bagchi S, Buckley S, Ettl M et al. (1998) Experience using the IBM supply chain simulator. In: Proceedings of the 1998 Winter Simulation Conference, Washington, DC, USA, pp. 1387–1394

Baglin G, Bruel O, Garreau A et al. (2001) Management industriel et logistique. Collection Gestion, Série: Productique et techniques quantitatives appliquées à la gestion. Economica, pp. 323–340

Banks J, Jain S, Buckley S et al. (2002) Opportunities for simulation in supply chain management. In: Proceedings of the 2002 Winter Simulation Conference, San Diego, California, pp. 1652–1658.

Bauer B, Odell J (2005) UML 2.0 and agents: How to build agent-based systems with the new UML standard. Eng. Appl. Artif. Intell. (Int. J.), 18:141–157

Brun A, Cavalieri S, Macchi M et al. (2002). Distributed simulation for supply chain coordination. In: Proceedings of the 12th International Working Seminar on Production Economics, Igls, Austria

Chehbi S, Derrouiche R, Ouzrout Y et al. (2003) Multi-agent supply chain architecture to optimize distributed decision-making. In: Proceedings of the 7th World Multi-conference on Systemics, Cybernetics and Informatics, SCI, 16, Orlando, USA

Chen F, Drezner Z, Ryan JK et al. (2000a) Quantifying the bullwhip effect in a simple supply chain: the impact of forecasting, lead times, and information. Manag. Sci., 46(3):436–443

Chen F, Drezner Z, Ryan J.K et al. (2000b) The impact of exponential smoothing forecasts on the bullwhip effect. Naval Res. Logist. (Int. J.), 47:269–286

Cloutier L, Frayret JM, D'Amours S et al. (2001) A commitment-oriented framework for networked manufacturing coordination. Int. J. Comput. Integr. Manuf., 14(6):522–534

Cohen MA, Lee HL (1988) Strategic analysis of integrated production distributed systems: models and methods. Oper. Res., 36(2):216–228

ECR (2006) http://www.ecrnet.org/

FIPA (2002) FIPA ACL message structure specification. Foundation for Intelligent Physical Agents. (On http://www.fipa.org/spcs/fipa00061/SC00061G.pdf)

Florea A (2002) Using Utility Values in Argument-based Negotiation. In: Proceedings of IC-AI'02, the 2002 International Conference on Artificial Intelligence, Las Vegas, Nevada, USA, pp. 1021–1026,

Fu Y, Souza R.D, Wu J (2000) Multi-agent enabled modeling and simulation towards collaborative inventory management in supply chain. In: Proceedings of the 2000 Winter Simulation Conference, Orlando, FL, USA, pp. 1763–1771

Fujimoto R (1999) Parallel and distributed simulation. In: Proceedings of the 1999 Winter Simulation Conference, Piscataway, NJ, USA, pp. 122–131

Giannoccaro I, Pontrandolfo P, Scozzi B (2003) A fuzzy echelon approach for inventory management in supply chains. Eur. J. Oper. Res., 149(1):185–196

Hieber R (2002) Supply chain management: a collaborative performance measurement approach. VDF

Hugos M (2003a) Essentials of supply chain management. John Wiley and Sons: 3–4

Hugos M (2003b) Essentials of supply chain management, John Wiley and Sons: 64–65

Jennings N, Faratin P, Lomuscio NR et al. (2001) Automated negotiation: Prospects, methods and challenges. Group. Decis. Sci. Nego. J., 10(2):199–215

John Taras CPIM (2009) http://www.johntaras.com/resume.htm

Kimbrough SO, Wu DJ, Zhong F (2002) Computers play the beer game: can artificial agents manage supply chains?. Decis. Support Syst., 33(3):323–333

Lee CB (2003) Multi-echelon inventory optimization. Evant White Paper Series

Lee HL, Padmanabhan V, Whang S (1997) The bullwhip effect in supply chains. Sloan Manag. Review, 38(3):93–102

Maturana F, Shen W, Norrie DH (1999) MetaMorph: an adaptive agent-based architecture for intelligent manufacturing. Int. J. Prod. Res., 37(10):2159-2174

McBurney P, Parsons S, Green J (2002) Forecasting market demand for new telecommunications services: an introduction. Telematics and Informatics, 19(3):225–249

Moyaux T, Chaib-draa B, D'Amours S (2004) An agent simulation model for the Quebec forest supply chain. In: Proceedings of the 8th International Workshop on Cooperative Information Agents (CIA), Erfurt, Germany, 3191: 226–241

Nfaoui EH, Ouzrout Y, El Beqqali O et al. (2006) An approach of agent-based distributed simulation for supply chains: Negotiation protocols between collaborative agents. In Proceedings of the 20th annual European Simulation and Modeling Conference, EUROSIS, Toulouse, France, pp. 290–295.

Parunak HVD 1996 Applications of distributed artificial intelligence in industry. In O'Hare GMP and Jennings NR (eds) Foundations of distributed artificial intelligence: pp. 71-76, John Wiley and Sons

Petersen SA, Divitini M, Matsken M (2001) An agent-based approach to modeling virtual enterprises. Prod. Plan. Control, 12(3):224–233

Pinedo M, Chao X (1999) Operations scheduling with applications in manufacturing and services. Mc Graw-Hill

Reutterer T, Kotzab HW (2000) The use of conjoint-analysis for measuring preferences in supply chain design. Ind. Mark. Manag., 29(1):27–35

Sadeh NM, Hildum DW, Kjenstad D et al. (2001) MASCOT: an agent-based architecture for dynamic supply chain creation and coordination in the Internet economy. Prod. Plan. Control, 12(3):212–223

Samii AK (2004) Stratégie logistique, supply chain management. Dunod, 3rd edition, pp. 14-15

Siala M, Campagne J, Ghedira K (2006) Proposition d'une nouvelle approche pour la gestion du disponible dans les chaînes logistiques. 6e Conférence Francophone de Modélisation et Simulation (MOSIM), Avril, Rabat, Maroc, pp. 412-421,

Simchi-Levi D, Kaminsky P, Simchi-Levi E (2000) Designing and managing the supply chain. McGraw-Hill Higher Education

Swaminathan JM, Smith SF, Sadeh NM (1998) Modeling supply chain dynamics: a multiagent approach. Decis. Sci., 29(3):607–632

Telle O, Thierry C., Bel G (2003) Simulation d'une relation client/fournisseur au sein d'une chaine logistique intégrée: mise en oeuvre industrielle. In: Proceedings of MOSIM03 (Conférence Francophone de MOdélisation et Simulation), Toulouse, France

Terzi S, Cavalieri S (2004) Simulation in the supply chain context: a survey. Comput. Ind. 53(1): 3–16

VICS Association (2009), http://www.vics.org/

Wu DJ (2001) Software agents for knowledge management: coordination in multi-agent supply chains and auctions. Expr. Syst. Appl., 20(1):51– 64

Zhao X, Xie J, Leung J (2002) The impact of forecasting model selection on the value of information sharing in a supply chain. Eur. J. Oper. Res., 142(2):321–344

Chapter 5
Web-service-based e-Collaborative Framework to Provide Production Control with Effective Outsourcing

A. Keshari[1], M. K. Tiwari[2] and R. Teti[3]

Abstract This chapter presents a conceptual framework of a web-services-based e-collaborative system to establish a real-time information-sharing platform for enabling collaboration among similar types of manufacturing firms. The main idea of this chapter is to integrate the process planning and scheduling activities with the proposed system considering outsourcing as a viable technique of enhancing machine utilization and system's performance. A mathematical model has been formulated that minimizes the broad objectives of machining cost and makespan. A multi-agent system constituting different functional and non-functional entities is considered to establish the collaboration among associated firms. A real-time information-sharing scenario has been proposed that monitors the existing shop-floor activities and accordingly generates the optimal or near-optimal process plan and schedule of job orders. In order to validate the robustness of the proposed system, extensive computational experiments are conducted over small and large problems. Results obtained are compared with each other and useful insights are drawn to establish the effectiveness of the same. Further, an illustrative example has been considered that demonstrate the working mechanism of the proposed framework.

Keywords Web-service, e-collaboration, production control, outsourcing

[1] A. Keshari
University of Naples Federico II, Italy

[2] M. K. Tiwari (✉)
Indian Institute of Technology, India
e-mail: mkt09@hotmail.com

[3] R. Teti
University of Naples Federico II, Italy

5.1 Introduction

Amidst the rapidly changing market scenario, products are more differentiated and refined. In order to survive in such an environment, manufacturers are required to focus on their core competencies and incorporate outsourcing for secondary activities. Outsourcing can be defined as the contracting out a part of the business process, such as, manufacturing, maintenance, accounting or logistics, to the partners (Kolisch, 2000). In the prevailing competitive environment, outsourcing acts as an effective measure for manufacturers to maximize their profit by contracting out the non-expertise operations to the efficient destinations. Outsourcing strategies represent a remunerative option when companies focus on future development prospects and intend to reduce the less-value-added activities. Owing to the functional and operational similarities, outsourcing within firms of similar type is more profitable and productive. Any firm that outsources an operation expects the following benefits:

1. exploration of its core competencies;
2. reduction in uncertainty through risk-pooling effect;
3. reduction in capital investments;
4. economic production;
5. improved utilization of the available resources and facilities;
6. enhancement of flexibility to deal with the varieties of product;
7. amelioration of reaction capability to changes in the customer demands;
8. enabling the firm to use supplier's technical knowledge to accelerate product development cycle time.

Firms share on-line information with other members of the group in collaboration, thus they can identify the most efficient means of outsourcing by comparing the statistics of each individual firm. This real-time information plays a key role in achieving high degree of manufacturing flexibility, minimization of stock resources and balanced load of the manufacturing machinery. Thus, industries are now looking forward to establish a potential collaborative information-sharing platform that assists in the grouping of firms (Innocenti and Labory, 2004). To ensure efficient outsourcing among members of a group, outsourcing decisions can be only taken after detailed analysis of the manufacturing activities and resources available at internal and external facilities. Therefore, it is expected that the platform must facilitate real-time information sharing in order to establish a coordinated inter-organizational planning and scheduling decisions.

This chapter presents a conceptual framework of web-services-based e-collaborative environment (WSC) where each collaborative member has been designed as a part of a multi-agent system (MAS) for facilitating automated information sharing. An information-sharing support system has been proposed to assess the real-time manufacturing capabilities of other group members. Also, it assists in drafting an economical production plan from the available information of internal manufacturing facilities and outsourcing activities. Specifically, the proposed framework provides an integration of process-planning and scheduling activities considering outsourcing opportunities. The process-planning module is responsible for the generation of an effective resource utilization plan. It encapsulates features of the part design specification, available machine characteristics and their mutual inter-relationships (Li *et al.,* 2002). In contrast, a scheduling module allocates proper tasks at the available resources on the shop-floor and manages overall flow of the production orders (Chung *et al.*, 2000). Thus, in order to control their production activities, distributed artificial intelligence methods have been utilized in the proposed system. Specifically, each individual firm has been designed as an MAS where different functional agents (machine agent (MA), task management agent (TMA), and production controlling agent (PCA)) are responsible for coordination and knowledge sharing. Moreover, in order to establish a seamless workflow plan, agents share their database and perform the corresponding manufacturing activities at diverse locations (internal manufacturing units and outsource manufacturing units). Simple-object access protocol (SOAP) is used as a communication channel between agents to achieve the interoperability of cross-platforms and widely distributed systems (Chan *et al.*, 2005).

The main objective of the proposed framework is to minimize the broad objectives of machining cost and makespan with the help of its coordination and control mechanism. A mathematical model has been formulated that includes an outsourcing aspect and its related intricacies during the working of the proposed system. Different problem scenarios of small and large size are generated to demonstrate the working of proposed framework, thereby validating its effectiveness.

The rest of the chapter is arranged in the following sequence. Sect. 5.2 summarizes the literature survey related to integration of scheduling and process planning. Sect. 5.3 details the constituting elements of proposed framework and explains their functioning as well. Sect. 5.4 describes the protocols used by the different functional agents. An illustrative example along with rigorous computational experiments is presented in Sect. 5.5. Finally, conclusions and future scope of research are presented in Sect. 5.6.

5.2 Literature Review

Over the last decade, outsourcing has been recognized as a key factor to ensure timely and reliable delivery of product and an effective technique to survive in rapidly changing competitive environment (Innocenti and Labory, 2004). It carries a paramount importance for manufacturing industries where, the main objective is to satisfy the customer demand before their due dates. It can be concluded from the comprehensive literature survey that a numerous research efforts have been focused in the broad domain of process planning and scheduling (Mamalis *et al.*, 1996; Khoshnevis, 2004).

Zijm (1995) and Kempenaers *et al.* (1996) presented a heuristic procedure to explore the alternative process plan for a given job order. Their suggested methodology attempts to identify the most efficient plan from multiple feasible process plans. Lee *et al.* (2001) presented two branch-and-fathoming algorithms to obtain optimal or near-optimal solutions for an operation-sequencing problem with the objective of minimizing the sum of machine, setup and tool-changing costs. Gologlu (2004) employed constraint-based operation sequencing for a knowledge-based process-planning problem. They adopted a four-level hierarchy comprising feature level, machining scheme level, operation level and tool level to attain a p-roper operations sequence.

Every manufacturing firm has its own limitations (resource limitation, limitation of order due date, technical limitation, facility limitation), which restrict the effectiveness of integrating process planning and scheduling. Therefore, an integrated approach is not sufficient to ensure full utility of available alternatives, i.e. internal and external facilities of production (Lee *et al.*, 2002). In order to handle these challenges, an effective method is to consider outsourcing aspects in the integrated process-planning and scheduling problems.

In the existing literature, Chung *et al.* (2000) developed an algorithm for a job-shop scheduling problem considering outsourcing and due dates as design constraints. They addressed the job-shop scheduling problem by solving a series of smaller sub-problems. Lee *et al.* (2002) addressed the advance planning and scheduling problem to meet the coustomer orders due dates. They asserted that planning and scheduling are inter-related issues and should be solved simulteneously with outsourcing. Norman *et al.* (1999) advocates for real-time information sharing and presented an overview of the agent-based architecture for a coordinated mixed-initiative supply chain. Their experimental analysis reveals that active-information sharing critically helps the development of coordination policies and plays a key role dynamically with finite capacity considerations. Similarly, applications of web-supported MASs in manufacturing problems have been reported in the literature (Huang *et al.*, 2000; Sikora and Shaw 1997; Ta and Yuehwern, 2001; Shen and Noorie, 1999; Petrie, 1996; Nahm and Ishikawa, 2005). Lux and Steminer (1995) examined the multi-agent framework and advocates that it models the problem in terms of the autonomous interacting individual agents and proves to be a more natural way of representing task allocation, team planning and user preference in an open environment. In the same vein, Tripathi *et al.* (2005) presented a multi-agent approach to solve the part selection and task allocation problem in flexible manufacturing systems.

The previous studies assume that pre-specified outsourcing zones and machines are already known to the manufacturer. However, the practical scenario differs where availability of resources at the outsource stations varies with the time. Therefore, it becomes imperative to consider the real-time information scenario that monitors the actual availability of resources and provides real-time data to the manufacturers. Therefore, considering the aforementioned research issues, a conceptual framework for integrated process-planning and scheduling activities with outsourcing opportunities has been designed with an MAS. The main objective of proposed framework is to provide automated online information integration among the loosely collaborating firms and thereby minimizes the makespan and machining cost of the job orders. The next section details the constituting elements of proposed framework and describes their implementation perspectives.

5.3 Design of Web-services-based e-Collaborative Framework

In this section, the working mechanism of the proposed WSC is presented followed by the detailed description of functional and non-functional agents. Thereafter, a mathematical model has been detailed in order to minimize the makespan and machining cost of the customer order.

5.3.1 The Proposed Framework

Basically, the e-collaboration among the firms can be defined as the collaboration among loosely coupled MASs of the associated firms that can communicate and share their ideas around the globe. In accordance to functional requirements of the intra-firm, each autonomous agent performs its functions independently and, has its own capability of planning, learning, decision-making, and execution activities. Also, they can communicate, coordinate and cooperate with other agents to share their knowledge, information and resources to avoid conflicts and to identify a best solution. The proposed model provides the collaboration among MASs, which comprises three types of functional agents: TMA (task management agent), PCA (production controlling agent), and MA (machine agent). Two types of servers are used: data acquisition server (DAS), and on-site collaborative server (OCS). Also, to maintain the information security and other safety measure for collaboration, it is facilitated with authentication server and universal description discovery and integration (UDDI) registration (Hung *et al.*, 2005).

The DAS acts as an interfacing server that plays an acquisition and transformation role of incoming and outgoing SOAP™ messages (http://www.w3c.org/tr/soap) respectively. It comprises two parts: web-service agent and data acquisition manager. A combined action of both establishes the SOAP communication interface for functional agents (PCA, MA) and thereby monitors the negotiation and routing of intra-firm information. The OCS is responsible for providing the message interfacing module for the TMA and thereby enables it to establish inter-firm and intra-firm communication facilities. Similar to the DAS, the OCS also consists of two components: web-service agent, and OCS communication manager; however, the interlinked construction between them is different.

The concerned functional agents are assigned some specific actions to monitor the manufacturing activities. Intra-firm control and planning of manufacturing activities are governed by PCA, whereas TMA is responsible for generating an effective schedule. When the firm receives a new production tasks, TMA decomposes these tasks into three sub-tasks:

1. core competency manufacturing operations;
2. manufacturing operations that can be performed within the firm as well as at outsource locations; and
3. fully outsourcing manufacturing operations.

For categories 2 and 3 of manufacturing operations, the host TMA sends a request message to the TMA of all collaborating firms to acquire the real-time information of available resources and machine facilities. In the same time frame, the host TMA also sends a request message to its PCA agent to search the available resources and machines of intra-firms. After receiving the response from collaborated TMA and its PCA, TMA defines the optimal schedule plan and assigns the corresponding tasks to available outsource resources.

5.3.2 Design of System Components of Model

5.3.2.1 Non-functional Agent (Web-service Agent)

A web service agent is an interfacing agent and performs various functions that are essential for establishing communication between WSC partners. Specifically, it is an inherent part of the defined servers (OCS, DAS), which are attached with the functional agents to setup a web-based communication network across each other. Web-service agent includes several functional mechanisms (Hung *et al.*, 2005) to fulfill the interfacing requirements and encapsulate the following key characteristics:

1. facilitates SOAP communication interface for messaging and coordination;
2. possesses a UDDI registry mechanism for registering and searching web services;
3. provides a security mechanism;
4. transforms communication messages in XML formats;
5. enables to access the local database;
6. provides a communication kernel, which is used to link all the encapsulated facilities of the web-service agent.

Each web-service agent has been linked with the WSC Internet network by initializing its local program. It establishes a virtual network among all collaborating web-service agents (Figs. 5.1 and 5.2). Thus, all virtually linked web-service agents can access each other over the Internet through SOAP communication. The web-service agent enables the working agent to work as remote system that can communicate with each collaborating system through the Internet. The web-service agent utilizes the popular mainstream-programming platform, Sun J2EE™ (http://java.sun.com).

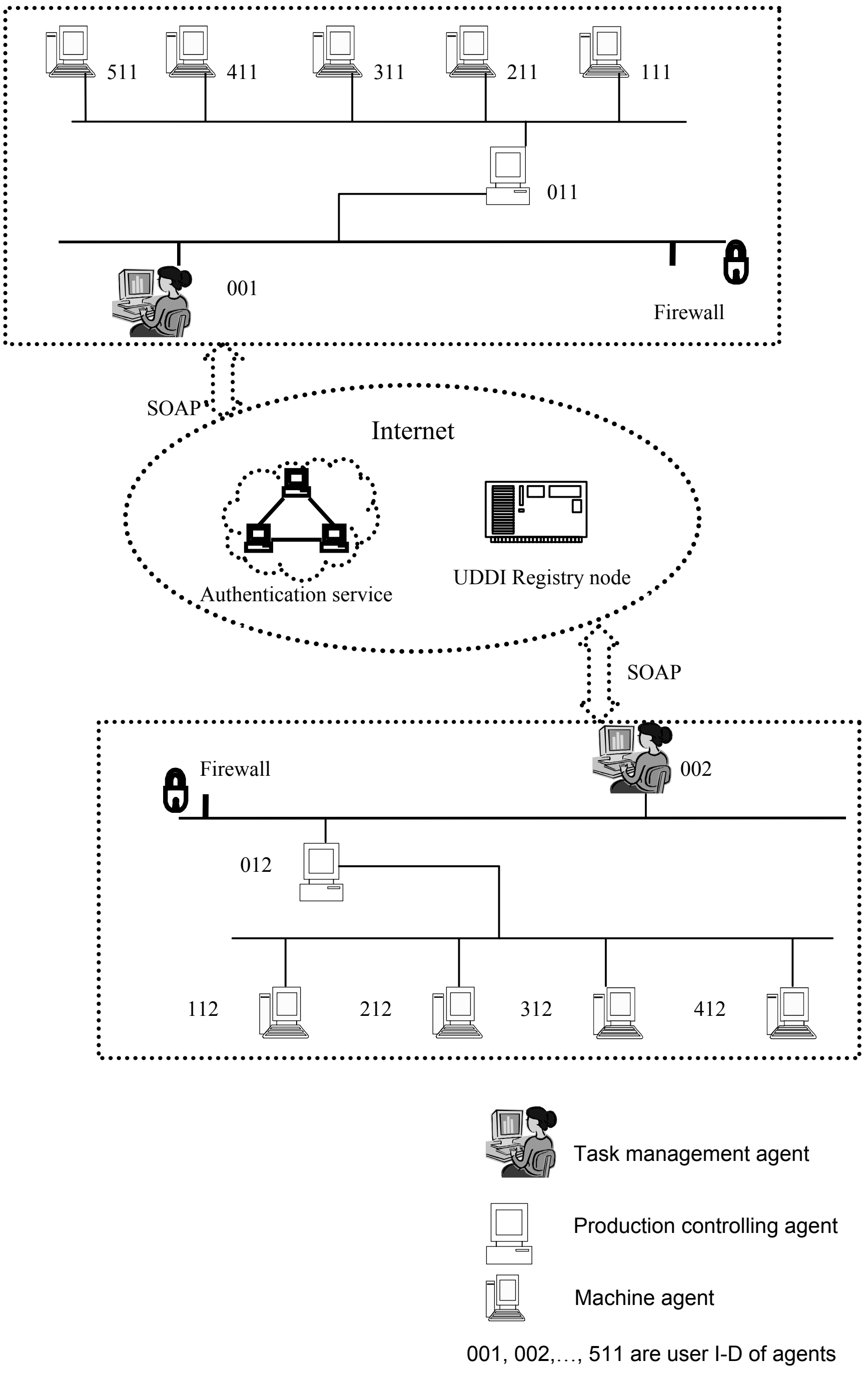


Fig. 5.1 An integrated framework of WSC network

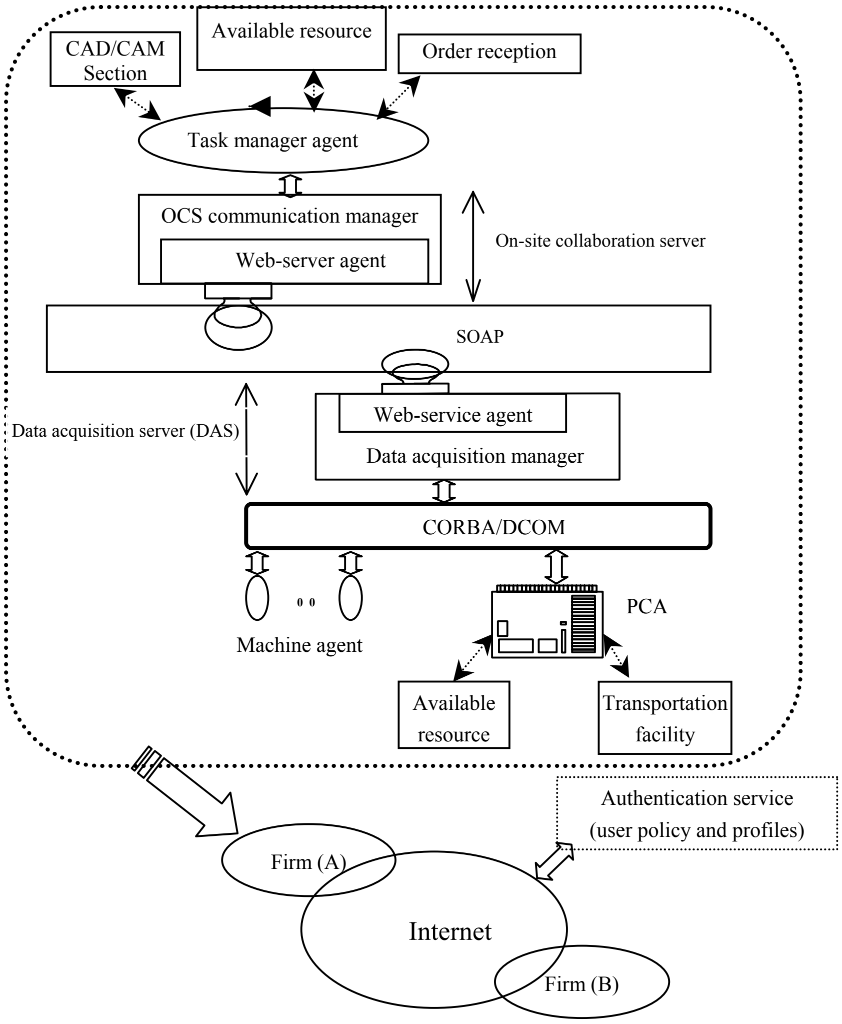


Fig. 5.2 Web services for the firms' collaboration

The internal structure of web-service agent is shown in Figure 5.3. The data acquisition manager and the OCS communication manager is inherently programmed with the web-service agent to construct the DAS and OCS respectively, and functionally works with the web-service agent to facilitate various functions included in it.

5.3.2.2 Functional Agent

Each functional agent (TMA, PCA and MA) participating in the MAS is a computing system that can autonomously react and reflexively respond to abrupt environmental changes. These agents can communicate over the Internet and their communication is regulated by a pre-defined defined protocol (Tripathi *et al.*, 2005). An agent's internal architecture is shown in Figure 5.4.

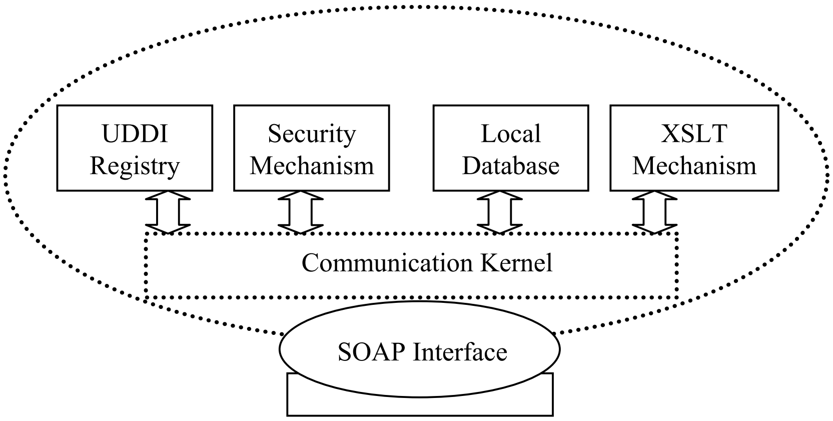


Fig. 5.3 Functional entities of web-service agent (Hung *et al.*, 2005)

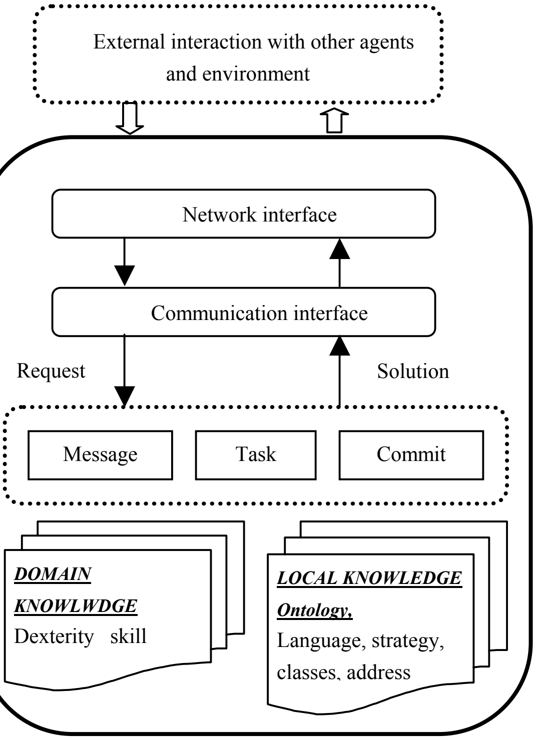


Fig. 5.4 Internal structure of a functional agent (Tripathi *et al.*, 2005)

Figure 5.4 comprises the following elements: a network interface, a local knowledge model, a domain knowledge model, and communication interface and task execution programs. On receiving the message, the agent proceeds as follows:

- encode the message content for further proceeding as the content of a particular task;
- execute the task; and
- return the solutions to the request maker.

Machine Agent

The MA is responsible for executing the part operations. It possesses all the information regarding the attributes of machine including its status, parts waiting in the buffer, capabilities and the tool facilities. Each MA has a unique identification number and a location. It starts executing a new process after it sends a completion report of previous work to the PCA, which in turn assigns the details of a new task to it.

Production Controlling Agent

It is responsible for carrying the production process according to the optimized schedule fetched by the TMA. It consists of all part-related information for each associated MA. The PCA can communicate with associated TMA, MAs and the PCAs of all other collaborating firms to deal with the outsourcing aspects of the manufacturing schedule. It monitors the allocation of tasks on the MAs, resource and transportation facilities. Moreover, the PCA receives the production plan from the associated TMA and stores it for the monitoring references.

Task Management Agent

Virtually, it acts as a managing agent and executes several activities, e.g. web-based communication between collaborating firms to assign outsource co-operation, defining operations for the products, management of resource, reception of coming orders and generation of an optimal schedule plan for its processing. In the development of an optimal scheduling plan, the parameters like processing time of the operation, transportation time, alternative machines, available tools, time available to perform the operation on the corresponding machines and precedence relationship between the operations are taken into account. The TMA agent utilizes several constraints and decision variables to simulate the real situation and precedence relationships between the operations. The mathematical model utilized by TMA in order to minimize the makespan and machining cost of the job orders are presented in next section.

5.3.3 The Mathematical Model

In this section, a mathematical model has been presented that integrates the process planning activity with the scheduling module for varying the number of parts to be manufactured using the proposed WSC system. Details of the notation and decision variables used throughout this chapter are as follows.

Notation

i:	part index ($i = 1, 2, ...,I$)
c_i:	number of part-type for ith part
j:	operation index ($j = 1, 2, \ldots, v$)
k:	machine index ($k = 1, 2, \ldots, m$)
θ_{ijk}:	processing time of operation j of part i on machine k
w_{ijk}:	total processing time for c_i part-types of part i
t:	tool index ($t = 1,2,..., T$)
UTC_t:	tool cost
TC_{ti}:	total tool cost for c_i part-types of part i
MCC:	machine change cost
MS:	makespan of the customer order
ϕ_{ijk}:	start-time of operation j of part i on machine k
$T_{k_1k_2}$:	transportation time between machine k_1 to machine k_2
STA_k:	starting time of availability of machine k
CTA_k:	closing time of availability of machine k

Decision Variables

$$p_{ijk} = \begin{cases} 1 & \text{if operation } j \text{ of part } i \text{ is assigned to machine } k \\ 0 & \text{otherwise,} \end{cases}$$

$$r_{j_1 j_2} = \begin{cases} 1 & \text{If } j_1 \text{ and } j_2 \text{ are the operation of same part} \\ & \text{and operation } j_1 \text{ precedes operation } j_2 \\ 0 & \text{otherwise,} \end{cases}$$

$$C_{ij}^{kk'} = \begin{cases} 1 & \text{if part } i \text{ is transferred from machine } k \text{ to} \\ & \text{machine } k' \text{ for operation } j \\ 0 & \text{otherwise,} \end{cases}$$

$$x_{jt} = \begin{cases} 1 & \text{if operation } j \text{ is executed using tool } t \\ 0 & \text{otherwise,} \end{cases}$$

$$y_k = \begin{cases} 1 & \text{if machine } k \text{ is selected} \\ 0 & \text{otherwise.} \end{cases}$$

Constraints

In order to reduce the problem complexities, each part-type of part *i* is processed simultaneously on machine *k* to perform the *j*th operation. The constraints related to the underlying problem have been classified into two categories: precedence relationships and non-overlapping constraints.

Precedence constraints: they deal with the pre-defined relationships between any two operations of the similar part. If operation j_1 precedes operation j_2 on part *i* then its precedence relationship can be mathematically modelled as:

$$\sum_k \left(p_{ij_1k}\varphi_{ij_1k} + p_{ij_1k}w_{ij_1k} + \sum_{k'} C_{ij}^{k'k} \; T_{k'k} \right) \le \sum_k p_{ij_2k}\varphi_{ij_2k} \, . \tag{5.1}$$

Non-overlapping constraints: these constraints are further classified on the basis of operations to be performed on machines and parts.

Non-overlapping Constraints for Operations to be Performed on the Same Machine

According to this constraint, the machine will start a new operation only after the completion of the previous operation. Let j_k be the set of operations that can be performed on the machine *k*; operation j_1 of part i_1 and operation j_2 of part i_2 are two such operations, which are to be performed on machine *k*. However, either j_1 may precede j_2 or vice versa will be true. Therefore, in order to distinguish this similarity, the following constraint has been modelled to ensure that the operation time of j_1 will not overlap with the operation time of j_2.

$$\begin{aligned} \left(\eta + p_{i_1j_1k}w_{i_1j_2k}\right) q_{j_1j_2k}^{i_1i_2} + \left(p_{i_2j_2k}\;\varphi_{i_2j_2k} - p_{i_1j_1k}\;\varphi_{i_1j_1k}\right) \\ \ge p_{i_1j_1k}w_{i_1j_1k} + \sum_{k'} C_{i_2j_2}^{k'k} \; T_{k'k} \end{aligned} \tag{5.2}$$

$$\left(\eta + p_{i_2 j_2 k} w_{i_2 j_2 k}\right)\left(1 - q_{j_1 j_2 k}^{i_1 i_2}\right) + \left(p_{i_1 j_1 k}\ \varphi_{i_1 j_1 k} - p_{i_2 j_2 k}\ \varphi_{i_2 j_2 k}\right) \qquad (5.3)$$
$$\geq p_{i_2 j_2 k} w_{i_2 j_2 k} + \sum_{k'} C_{i_1 j_1}^{k'k}\ T_{k'k}$$

where η is a very large positive number. Constraints 5.2 and 5.3 will be effective only when operation j_1 and operation j_2 have been performed on the same machine. It can be expressed as

$$p_{i_1 j_1 k} \times p_{i_2 j_2 k} = 1. \qquad (5.4)$$

Non-overlapping Constraints for Operations of Same Part

This constraint is valid for those operations that have no precedence relationships with others. Accordingly, it checks the feasibility of the process by restricting the other operations to be performed in the non-overlapping time period of the similar part.

$$\eta\left(1 - r_{j_1 j_2}\right) + \sum_k \left(p_{ij_2k}\ \varphi_{ij_2k} - \sum_{k'} C_{ij_2}^{k'k}\ T_{k'k} - p_{ij_1k}\ \varphi_{ij_1k}\right) - \sum_k p_{ij_1k}\ \varphi_{ij_1k} \geq 0 \quad (5.5)$$

$$\eta\left(r_{j_1 j_2}\right) + \sum_k \left(p_{ij_1k}\ \varphi_{ij_1k} - \sum_{k'} C_{ij_1}^{k'k}\ T_{k'k} - p_{ij_2k}\ \varphi_{ij_2k}\right) - \sum_k p_{ij_2k}\ \varphi_{ij_2k} \geq 0 \quad (5.6)$$

Process plan selection constraint: this constraint implies that one and only one machine is selected for an operation.

$$\sum_k p_{ijk} = 1. \qquad (5.7)$$

Equation 5.8 calculates the total processing time for c_i part-types of part i while Equation 5.9 determines the total cost of using tools for all part-types of a particular part. Constraints 5.10, 5.11, 5.12, 5.13 and 5.14 deal with the starting and closing time of machine availability.

$$\mathrm{w}_{ijk} = c_i\ \theta_{ijk}, \qquad (5.8)$$

$$TC_{ti} = UTC_t \cdot c_i, \qquad (5.9)$$

$$\phi_{ijk} \geq 0, \qquad (5.10)$$

$$STA_k \geq 0, \tag{5.11}$$

$$CTA_k \geq 0, \tag{5.12}$$

$$STA_k \leq CTA_k, \tag{5.13}$$

$$STA_k \leq \phi_{ijk} \leq CTA_k. \tag{5.14}$$

5.3.4 Overall Objective Function

The overall objective function of the TMA is to minimize the makespan and machining cost of the job orders and thereby schedules the production plan by considering the outsourcing activities. The makespan objective can be mathematically formulated as:

$$MS = \sum_i \sum_j \sum_k \left(p_{ijk} \ \varphi_{ijk} + p_{ijk} w_{ijk} + \sum_{k'} C_{ij}^{k'k} \ T_{k'k} \right). \tag{5.15}$$

The objective of machining cost conceived in this chapter constitutes tool cost, machine cost and machine changing cost.

Tool Cost (*OTC*)

It refers to the cost of using tools for all operations to be performed on a part-type.

$$OTC = \sum_j \sum_t TC_t \cdot x_{jt}, \tag{5.16}$$

Machine Cost (*MC*)

It is the total cost of using particular machines for performing different operations of a part-type. It can be mathematically represented as

$$MC = \sum_{k=1}^{m} MC_k \cdot y_k. \tag{5.17}$$

where MC_k is the machine cost index and it is a constant for a particular machine.

Machine Change Cost (*MCC*)

When adjacent operations are performed on different machines, there is a need for machine change. The cost incurred due to machine change can be represented by:

$$MCC = MC_k \times \sum_{j=1}^{J} \psi\left(M_{k+1}, M_k\right) \tag{5.18}$$

where M_k is the machine identification,

$$\psi\left(a,b\right) = \begin{cases} 1 & \text{if } a \neq b \\ 0 & \text{otherwise,} \end{cases} \tag{5.19}$$

Therefore, total machining cost (*TMC*) can be calculated as the sum of 5.16, 5.17 and 5.18:

$$TMC = OTC + MC + MCC. \tag{5.20}$$

Since the cost and time objectives are measured in different units, a normalization approach has been adopted to integrate these objectives and finally formulate a normalized overall objective function. The normalized approach can be defined as:

$$\text{norm}\left(x\right) = \frac{x_i - x_l}{x_u - x_l}$$

where, x_i is the current counter of x at ith iteration, x_l is its lower bound and x_u is its upper bound. Following this strategy, normalized objectives of machining cost (norm (*TMC*)) and makespan (norm (*MS*)) have been calculated. Therefore, the overall objective function (*OBJ*) can be written as:

$$\text{Minimize } OBJ = \text{norm}\left(TMC\right) + \text{norm}\left(MS\right). \tag{5.21}$$

5.4 Protocols of Functional Agents

5.4.1 Machine Agent Protocol

The main role of MA protocol is to execute the commands of the PCA at each event (starting and completion of operation) and accordingly respond to the request sent by the PCA. Moreover, depending on the type of occurring events, the

PCA updates its database and directs the MA to execute the subsequent operations. A diagrammatic view of the MA protocol is shown in Figure 5.5.

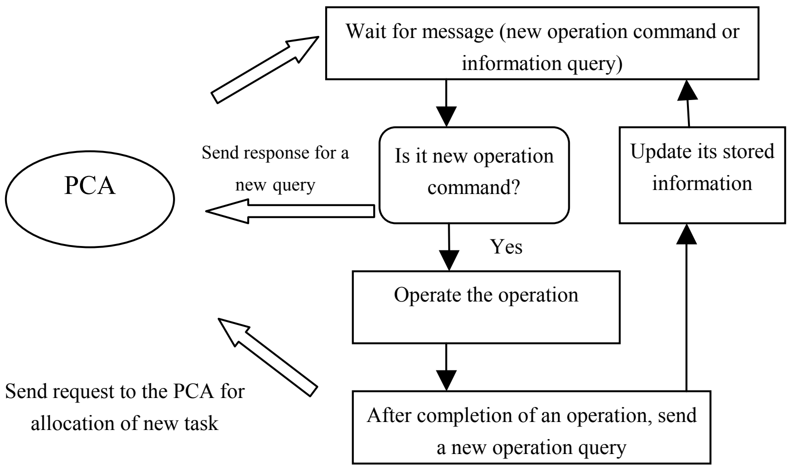

Wait for message (new operation command or information query)
PCA
Send response for a new query
Is it new operation command?
Update its stored information
Yes
Operate the operation
Send request to the PCA for allocation of new task
After completion of an operation, send a new operation query

Fig. 5.5 Machine agent protocol

5.4.2 Production Controlling Agent Protocol

The protocol of the PCA (Figure 5.6) guides the PCA to establish proper negotiations with other associated agents such as TMA, MAs, and the PCA of collaborating firms. Thus, it ensures coordination among several operations, e.g. event-wise communication between PCA and associated MA, updating real-time information of all individual MAs and accordingly allocating tasks to the MAs, etc.

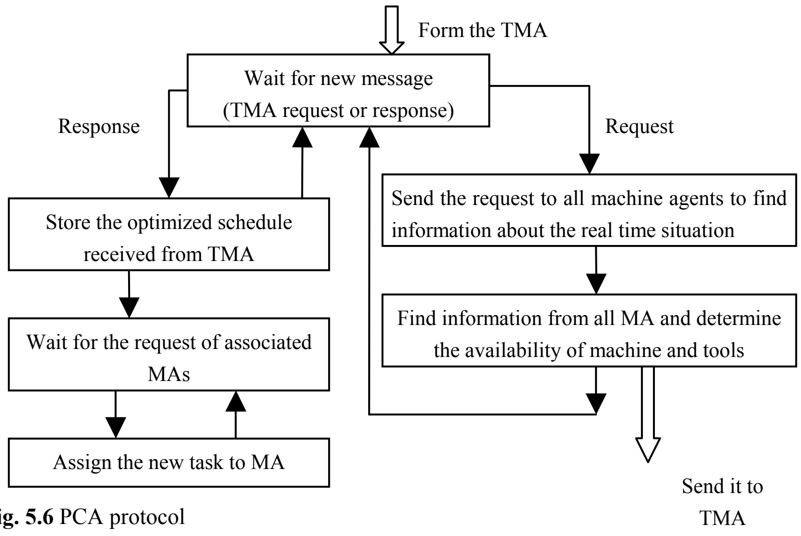

Form the TMA
Wait for new message (TMA request or response)
Response
Request
Store the optimized schedule received from TMA
Send the request to all machine agents to find information about the real time situation
Find information from all MA and determine the availability of machine and tools
Wait for the request of associated MAs
Assign the new task to MA
Send it to TMA

Fig. 5.6 PCA protocol

5.4.3 Task Management Agent Protocol

At the arrival of new job orders, the TMA protocol identifies the number of operations required to be performed and accordingly classifies those operations into internal and external ones. Conceptually, it sends a request message to collaborating PCA and TMAs for sharing the information regarding processing time, available alternative machines and tools from their database. In response, the PCA provides required information on the basis of the existing scenario of MAs (time to be idle). Thus, after identifying all the desired information, the TMA evaluates an optimum schedule plan and tooling arrangements. The diagram of the TMA protocol is shown in Figure 5.7.

5.4.4 Hierarchy of Steps in Web-service-based e-collaborative System

The steps of message processing are as follows:

Step 1: Provides new order information related to part and part-type to the host TMA.
Step 2: Define the number of operations and its precedence relationship.
Step 3: Decompose defined operations into the three sub-tasks discussed in Sect. 5.3.1.
Step 4: Request message transmission, e.g. from host TMA to collaborating TMA, message: ((b) & (c) type sub-task description); TMA to the associated PCA, message: ((a) & (b) type sub-task description).
Step 5: Response transmission from associated PCA to the host TMA.
Step 6: Response transmission from collaborating outsourcing TMAs to the host TMA. Response message contains processing time of the operations, staring time for the availability of machines, and available tools.
Step 7: Generation of optimized scheduling plan by TMA on the basis of obtained information.
Step 8: Message transmission from host TMA to outsource TMA to assign the allocated outsource task.
Step 9: Transmission of optimum schedule plan from TMA to associated PCA.
Step 10: Monitor the processing of operations along with the defined schedule.

5.5 Results and Discussion

In order to authenticate the proposed model, different problem instances of varying size have been considered over a time horizon. It has been assumed that there are two manufacturing firms A and B and both are exploring the various measures to improve the productivity and enhance the customer satisfaction level. Owing to achieve their desired objectives, firms are promoting outsourcing aspects among each other by sharing individual databases with the help of different functional agents employed in the proposed WSC network. The shop-floor scenario considered in this chapter is similar to that of Lee *et al.* (2002). The details of the dataset generated for simulation purposes are as follows.

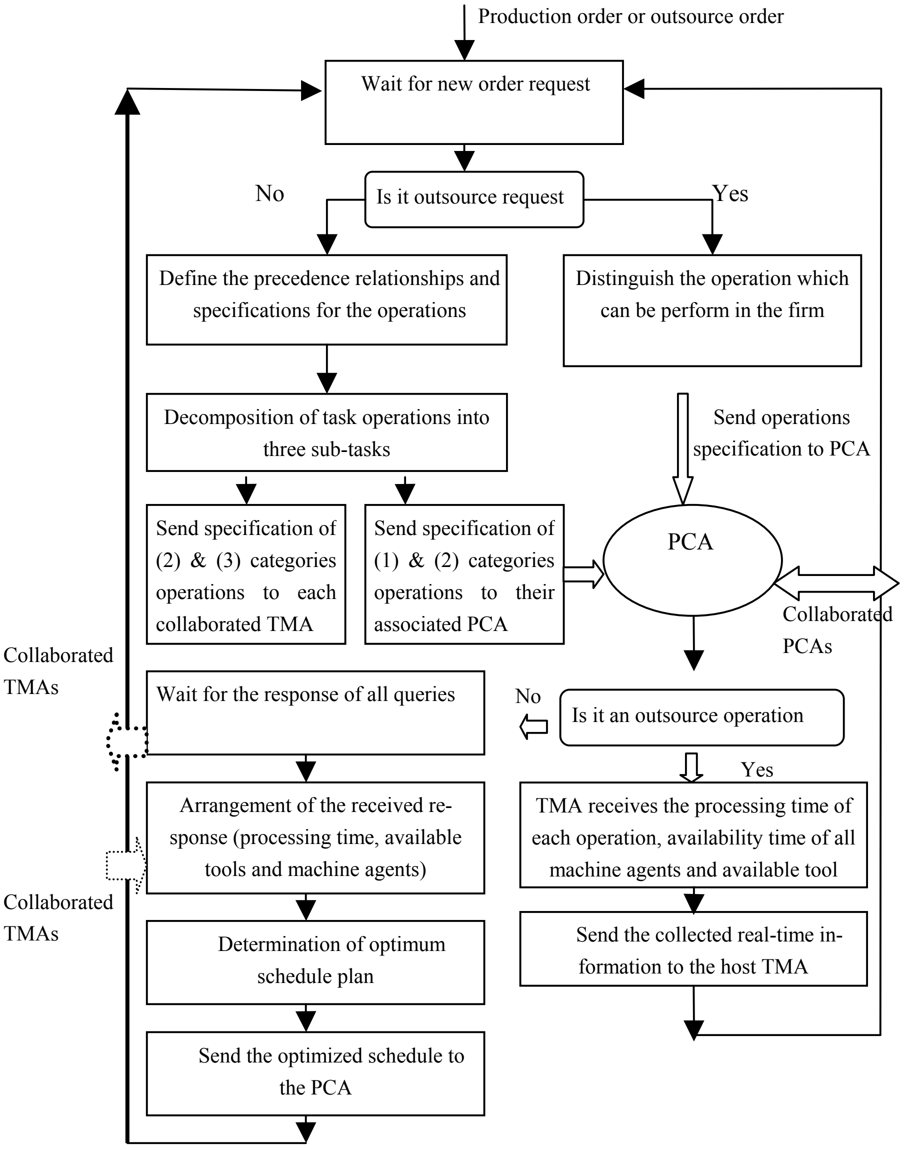

Production order or outsource order
Wait for new order request
No
Is it outsource request
Yes
Define the precedence relationships and specifications for the operations
Distinguish the operation which can be perform in the firm
Decomposition of task operations into three sub-tasks
Send operations specification to PCA
Send specification of (2) & (3) categories operations to each collaborated TMA
Send specification of (1) & (2) categories operations to their associated PCA
PCA
Collaborated PCAs
Collaborated TMAs
Wait for the response of all queries
No
Is it an outsource operation
Yes
Arrangement of the received response (processing time, available tools and machine agents)
TMA receives the processing time of each operation, availability time of all machine agents and available tool
Collaborated TMAs
Determination of optimum schedule plan
Send the collected real-time information to the host TMA
Send the optimized schedule to the PCA

Fig. 5.7 MA protocol

Manufacturing firm A consists of five MAs: M_1, M_2, M_3, M_4, M_5; firm B can outsource MAs M_6, M_7 and M_8 to firm A. Five parts are required to be manufactured in different quantities by firm A. A fixed number of operations are required to be performed for a particular part which have specific precedence relationships and other related constraints. Also, every operation is assigned some alternative machines and tools from the available pool of resources. The details of available machines and tools for all the operations are detailed in Table 5.1.

Table 5.1 List of tools and machines available for different operations

List of operations	Possible alternative machines	Possible alternative tools
O_1	M_1, M_2, M_3	T_1, T_2, T_3
O_2	M_2, M_3	T_4, T_6
O_3	M_1, M_2	T_1, T_2, T_3
O_4	M_3, M_4	T_8, T_2, T_9
O_5	M_2, M_3, M_5	T_1, T_2, T_3
O_6	M_1, M_2, M_5	T_1, T_2
O_7	M_5, M_3, M_4	T_9, T_3
O_8	M_2, M_1, M_4	T_1, T_3
O_9	M_1, M_5	T_7, T_4
O_{10}	M_2, M_3	T_9, T_8
O_{11}	M_4, M_3, M_5	T_5, T_4
O_{12}	M_2, M_1, M_5	T_7, T_6
O_{13}	M_6, M_7, M_8	T_4, T_5, T_6
O_{14}	M_7, M_8	T_3, T_8, T_7
O_{15}	M_7, M_8, M_6	T_3, T_5, T_1

Altogether, there are 15 operations to be performed for five parts. The precedence relationship among these operations can be represented as:

Part 1	Part 2	Part 3	Part 4	Part 5
$O_1 \rightarrow O_2 \rightarrow O_3$	$O_4 \rightarrow O_5$	$O_6, O_7 \rightarrow O_8$	$O_9 \rightarrow O_{10} \rightarrow O_{11} \rightarrow O_{12}$	$O_{13} \rightarrow O_{14} \rightarrow O_{15}$

The proposed web-based network is employed within both the firms and based on the protocols of different functional and non-functional agents; operations of all the parts are categorized into three categories, which are mentioned earlier in Sect. 5.1. Based on this classification, it has been found that firm A has expertise to perform the operations O_1, O_2, O_4, O_5, O_6, O_7, O_8, O_{11}, and O_{12}, and they can be processed within the firm. However, operations O_3, O_9, and O_{10} are non-expertise operations and can be performed within the firm with large processing time as compared to the expertise outsourcing locations. Thus, it is desirable to outsource these operations to diverse locations for reducing the processing time and cost. Operations O_{13}, O_{14}, and O_{15} can be performed at outsource locations only due to the unavailability of machines and tools within the native firm.

The processing time for all expertise operations of a part-type is randomly generated in the range of 0.5–1.0 hours, whereas it has been assumed that non-expertise operations take 1.5 times normal processing time. It has been considered that transportation time between internal manufacturing units is negligible in comparison to the processing time and outsource transportation time. The pre-defined scenario considered in this chapter assumes that all possible outsourcing MAs and tools are already available on the shop-floor. In contrast, the proposed real-time information-sharing based outsourcing scenario assumes that processing time of each operation, and availability of machine and tool are utilized for determining a schedule plan. In order to solve the underlying problem for different instances generated here, we have used the sample-sort simulated annealing (SSA) algorithm due to its out-performing nature in handling such problems. A brief overview of SSA is detailed in the following section.

5.5.1 Sample-sort Simulated Annealing Algorithm

SSA is a recently introduced random search technique that has been developed by Thompson and Bilbro (2005). It is an improved form of simulated annealing algorithm, whose optimization procedure is analogous to the mode of physical annealing given by Kirkpatrick *et al.* (1983). The virtue of SSA lies in its serialization and parallelism properties, where multiple *n* samplers propagate simultaneously parallel to each other in search of the optimal solution. In guidance of some operators, these *n* samplers interact with each other to evaluate and update their solutions. The combined search procedure of these samples enables the algorithm to explore a very large search space in minimum computational time. It is a problem-independent algorithm that preferably suits the scheduling problem irrespective of combinatorial hardness (Shukla *et al.*, 2007).

5.5.2 Experimental Analysis

The main idea of performing experimental analysis is to visualize the effect of real-time information sharing and thereby validate the use of same in the proposed framework. Two shop-floor scenarios are considered where firm A receives two types of part order, namely four parts and five parts. For each part order, extensive computational experiments are conducted to reveal the significance of impact of using machines and tools by different operations. Therefore, in the case of real-time information sharing, machine and tool availability are varied, whereas it has been kept constant for a fixed pre-defined dataset. This situation is considered for both types of job orders and the results are obtained by implementing the SSA algorithm in the problem under consideration. The values of machining cost, makespan and normalized objective function for four- and five-part orders under real-time information and fixed pre-defined dataset are listed in Table 5.2 and Tables 5.3 and 5.4 respectively. It can be clearly seen from these results that the makespan value is always found to be better in case of the real-time scenario as compared to fixed one. A significant difference has been witnessed in the results of makespan, machining cost and overall objective function values for four- and five-part orders in the real-time scenario, thus signifying the importance of using the real-time scenario in the proposed framework.

Table 5.2 Simulation results for four-part problem under different scenarios

Problem instance	Part	No. of part types	Fixed data set			
			Makespan	Machining cost	Normalized overall objective	Starting of the availability of machine
1	1	14	40.497	3108	0.714778	3, 5, 0.5, 6, 6.5, 0, 0, 0
	2	9				
	3	10				
	4	6				
2	1	8	41.672	3108	0.747556	2, 1.5, 0, 0, 1.5, 0, 0, 0
	2	8				
	3	9				
	4	12				
3	1	12	36.728	2910	0.608111	0, 0.5, 1, 0.5, 1.5, 0, 0, 0
	2	7				
	3	13				
	4	9				
4	1	12	49.572	2937	0.965111	1, 0.5, 2.5, 2.5, 5, 0, 0, 0
	2	8				
	3	13				
	4	15				
5	1	16	49.369	2969	0.959889	6.5, 5, 8, 7, 4, 0, 0, 0
	2	10				
	3	14				
	4	12				

Table 5.2 (continued)

Problem Instance	Part	No. of Part types	Proposed real-time information sharing concept			
			Makespan	Machining cost	Normalized overall objective	Starting of the availability of machine
1	1	14	34.54	3108	0.549556	3, 5, 0.5, 6, 6.5, 1.3, 0, 2.6
	2	9				
	3	10				
	4	6				
2	1	8	37.73	2942	0.636222	2, 1.5, 0, 1.5, 0, 2.5, 0.5, 1
	2	8				
	3	9				
	4	12				
3	1	12	28.91	2950	0.391333	0, 0.5, 1, 0.5, 1.5, 2.5, 3.5, 2.5
	2	7				
	3	13				
	4	9				
4	1	12	45.43	2890	0.849556	1, 0.5, 2.5, 2.5, 5, 4, 2.5, 3.5
	2	8				
	3	13				
	4	15				
5	1	16	41.62	2953	0.744444	6.5, 5, 8, 7, 4, 2.5, 2.5, 2
	2	10				
	3	14				
	4	12				

Table 5.3 Simulation results for five-part problem under different scenarios

Problem instance	Part	No. of part types	Fixed data set			
			Makespan	Machining cost	Normalized overall objective	Starting of the availability of machine
1	1	8				2, 0, 0.5, 1, 1.5, 0, 0, 0
	2	6				
	3	8	28.26	3885	0.2454	
	4	6				
	5	8				
2	1	10				1, 0, 2, 0, 0, 0, 0, 0
	2	10				
	3	12	42.28	3885	0.596	
	4	11				
	5	7				

Table 5.3 (continued)

Problem instance	Part	No. of part types	Proposed real-time information sharing concept			
			Makespan	Machining cost	Normalized overall objective	Starting of the availability of machine
3	1	10	41.11	3630	0.5642	2, 0.5, 4.5, 2, 0.5, 0, 0, 0
	2	8				
	3	12				
	4	10				
	5	7				
4	1	11	56.55	3601	0.9498	2, 2.5, 1.5, 3.5, 2, 0, 0, 0
	2	7				
	3	15				
	4	15				
	5	10				
5	1	10	54.32	3721	0.8952	2.5, 2.5, 4, 2, 3, 0, 0, 0
	2	7				
	3	12				
	4	15				
	5	9				

Table 5.4 Simulation results for five-part problem under different scenarios

Problem instance	Part	No. of part types	Proposed real-time information sharing concept			
			Makespan	Machining cost	Normalized overall objective	Starting of the availability of machine
1	1	8	22.31	3885	0.0967	2, 0, 0.5, 1, 1.5, 0, 4.5, 0
	2	6				
	3	8				
	4	6				
	5	8				
2	1	10	38.82	3671	0.5072	1, 0, 2, 0, 0, 3, 3, 1
	2	10				
	3	12				
	4	11				
	5	7				
3	1	10	33.11	3644	0.3641	2, 0.5, 4.5, 2, 0.5, 2, 1, 0
	2	8				
	3	12				
	4	10				
	5	7				

Table 5.4 (continued)

Problem instance	Part	No. of part types	Proposed real-time information sharing concept			
			Makespan	Machining cost	Normalized overall objective	Starting of the availability of machine
4	1	11				2, 2.5, 1.5,
	2	7				3.5, 2, 4,
	3	15	46.89	3654	0.7089	1.5, 0
	4	15				
	5	10				
5	1	10				2.5, 2.5, 4,
	2	7				2, 3, 0,
	3	12	43.59	3664	0.6265	0.5, 6
	4	15				
	5	9				

The operation sequence, corresponding selected machines, tools and processing time of each operation for a four-part order under fixed dataset scenario are detailed in Table 5.5. The results for the real-time scenario portraying similar information have been listed in Table 5.6. Similar results to those in Tables 5.5 and 5.6 are obtained for a five-part order under different situations and listed in Tables 5.7 and 5.8. The results obtained imply that the number of outsource operations varies with the scenario under consideration. It has been found that the real-time information scenario proves to be an efficient way of minimizing the makespan and machining cost. Based on the availability of resources present in other firms, the proposed system identifies the best process plan and accordingly schedules the jobs considering the real-time information scenario.

Table 5.5 Job order of four parts using fixed dataset

Problem instance 1

Operation sequence	O_9	O_{10}	O_1	O_2	O_4	O_3	O_{11}	O_6	O_7	O_5	O_{12}	O_8
Machine	M_6	M_7	M_1	M_3	M_4	M_8	M_3	M_1	M_4	M_2	M_1	M_4
Tool	T_7	T_8	T_1	T_4	T_9	T_1	T_5	T_1	T_3	T_2	T_6	T_3
Processing time	3.85	5.49	10.89	12.78	8.08	10.32	4.57	7.83	9.61	10.49	4.86	10.18

Problem instance 2

Operation sequence	O_1	O_9	O_2	O_7	O_3	O_{10}	O_{11}	O_8	O_4	O_5	O_6	O_{12}
Machine	M_2	M_8	M_3	M_3	M_6	M_8	M_5	M_4	M_3	M_2	M_1	M_5
Tool	T_3	T_7	T_6	T_9	T_3	T_9	T_5	T_1	T_8	T_2	T_2	T_6
Processing time	6.58	10.45	5.89	11.11	6.84	7.59	12.13	10.86	9.78	5.61	6.21	7.99

Problem instance 3

Operation sequence	O_9	O_1	O_{10}	O_6	O_{11}	O_2	O_4	O_3	O_7	O_5	O_{12}	O_8
Machine	M_8	M_2	M_8	M_1	M_5	M_2	M_4	M_8	M_3	M_2	M_5	M_4
Tool	T_4	T_2	T_9	T_2	T_4	T_6	T_2	T_1	T_3	T_2	T_6	T_3
Processing time	9.60	9.63	7.80	9.84	7.04	14.58	8.36	8.49	11.12	6.02	8.77	14.58

Table 5.5 (continued)

Problem instance 4

Operation sequence	O_9	O_{10}	O_1	O_4	O_{11}	O_{12}	O_6	O_7	O_2	O_3	O_5	O_8
Machine	M_8	M_7	M_3	M_3	M_5	M_5	M_1	M_4	M_2	M_7	M_2	M_1
Tool	T_7	T_9	T_2	T_8	T_4	T_6	T_2	T_3	T_6	T_2	T_1	T_1
Processing time	10.37	11.37	14.07	8.92	13.72	10.59	18	16.60	8.64	10.81	6.70	11.19

Problem instance 5

Operation sequence	O_7	O_9	O_4	O_1	O_6	O_8	O_2	O_{10}	O_3	O_5	O_{11}	O_{12}
Machine	M_5	M_7	M_4	M_3	M_2	M_1	M_2	M_7	M_2	M_5	M_5	M_1
Tool	T_3	T_7	T_8	T_2	T_3	T_2	T_6	T_9	T_1	T_2	T_4	T_6
Processing time	10.57	8.25	8.4	10.59	8.98	15.47	14.32	13.26	16.4	12.3	12.69	8.90

Table 5.6 Job order of four-parts using real-time information

Problem instance 1

Operation sequence	O_1	O_2	O_3	O_4	O_5	O_9	O_7	O_8	O_{10}	O_{11}	O_{12}	O_6
Machine	M_1	M_2	M_7	M_3	M_3	M_8	M_5	M_1	M_6	M_4	M_1	M_2
Tool	T_3	T_4	T_2	T_8	T_3	T_7	T_9	T_1	T_9	T_4	T_6	T_2
Processing time	11.48	9.14	7.42	7.2	4.95	5.78	8.6	8.74	3.97	4.39	5.64	6.31

Problem instance 2

Operation sequence	O_9	O_1	O_7	O_6	O_{10}	O_2	O_8	O_4	O_5	O_{11}	O_3	O_{12}
Machine	M_8	M_1	M_3	M_3	M_2	M_1	M_1	M_4	M_2	M_3	M_1	M_2
Tool	T_7	T_2	T_3	T_2	T_8	T_6	T_1	T_8	T_2	T_4	T_1	T_6
Processing time	8.50	7.16	4.76	4.57	8.56	7.66	5.82	4.52	6.23	10.10	6.19	6.04

Problem instance 3

Operation sequence	O_4	O_7	O_6	O_9	O_{10}	O_5	O_2	O_8	O_{10}	O_3	O_{11}	O_{12}
Machine	M_3	M_4	M_5	M_7	M_1	M_2	M_3	M_4	M_7	M_2	M_5	M_2
Tool	T_2	T_9	T_2	T_7	T_1	T_2	T_6	T_1	T_9	T_1	T_4	T_7
Processing time	6.46	10.60	11.38	5.73	6.31	3.97	8.74	12.33	4.98	7.57	4.93	5.13

Problem instance 4

Operation sequence	O_9	O_1	O_{10}	O_2	O_4	O_6	O_7	O_8	O_3	O_{11}	O_5	O_{12}
Machine	M_8	M_3	M_7	M_3	M_3	M_1	M4	M_4	M_1	M_4	M_2	M_5
Tool	T_7	T_1	T_9	T_6	T_2	T_2	T_3	T_3	T_2	T_4	T_2	T_6
Processing time	10.74	8.07	7.75	9.73	7.89	10.30	7.24	12.42	7.05	9.21	5.68	10.72

Problem instance 5

Operation sequence	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8	O_9	O_{10}	O_{11}	O_{12}
Machine	M_8	M_3	M_7	M_3	M_3	M_1	M_4	M_4	M_1	M_4	M_2	M_5
Tool	T_1	T_6	T_2	T_2	T_1	T_1	T_3	T_1	T_7	T_8	T_5	T_6
Processing time	11.6	9.50	11.76	8.14	8.43	11.04	8.55	11.21	9.29	10.41	8.96	7.44

Table 5.7 Job order of five parts using fixed dataset

Problem instance 1

Operation sequence	O_{13}	O_9	O_{14}	O_4	O_1	O_2	O_3	O_{10}	O_7	O_5	O_{11}	O_6	O_8	O_{12}	O_{15}
Machine	M_7	M_6	M_7	M_3	M_2	M_2	M_7	M_8	M_5	M_5	M_4	M_1	M_1	M_1	M_8
Tool	T_4	T_4	T_3	T_2	T_2	T_4	T_1	T_8	T_9	T_3	T_5	T_1	T_3	T_6	T_1
Processing time	9.89	4.15	5.96	4.71	6.6	5.49	5.69	4.02	5.97	6.60	6.19	5.92	9.08	5.1	5.36

Problem instance 2

Operation sequence	O_1	O_{13}	O_2	O_4	O_9	O_6	O_7	O_8	O_{10}	O_5	O_{11}	O_{12}	O_3	O_{14}	O_{15}
Machine	M_3	M_6	M_3	M_4	M_8	M_5	M_5	M_2	M_7	M_2	M_5	M_5	M_1	M_7	M_6
Tool	T_2	T_6	T_6	T_8	T_7	T_2	T_9	T_3	T_8	T_1	T_5	T_6	T_2	T_7	T_3
Processing time	11.23	8.37	9.62	11.42	10.63	9.69	7.59	10.11	12.84	12.12	8.04	7.26	6.4	8.62	8.22

Problem instance 3

Operation sequence	O_9	O_{10}	O_1	O_2	O_7	O_{13}	O_4	O_5	O_6	O_{11}	O_{14}	O_3	O_{15}	O_8	O_{12}
Machine	M_8	M_6	M_1	M_2	M_3	M_6	M_4	M_1	M_5	M_3	M_8	M_2	M_6	M_1	M_2
Tool	T_7	T_8	T_2	T_6	T_3	T_6	T_2	T_2	T_2	T_5	T_8	T_1	T_1	T_3	T_6
Processing time	7.75	8.66	11.57	8.26	14.59	5.23	6.52	11.37	9.00	6.66	8.23	9.62	8.54	9.31	7.52

Problem instance 4

Operation sequence	O_1	O_2	O_4	O_7	O_5	O_6	O_8	O_3	O_9	O_{10}	O_{11}	O_{12}	O_{13}	O_{14}	O_{15}
Machine	M_3	M_3	M_4	M4	M_2	M_5	M_2	M_7	M_6	M_6	M_4	M_5	M_6	M_8	M_6
Tool	T_3	T_6	T_2	T_9	T_2	T_1	T_1	T_2	T_7	T_9	T_5	T_6	T_6	T_7	T_1
Processing time	9.38	7.66	8.40	13.98	6.12	16.5	14.11	11.77	15.24	10.60	17.04	10.17	6.87	7.11	7.55

Problem instance 5

Operation sequence	O_1	O_9	O_7	O_2	O_{10}	O_8	O_4	O_3	O_{13}	O_{11}	O_6	O_{12}	O_{14}	O_5	O_{15}
Machine	M_3	M_7	M_5	M_2	M_7	M_4	M_4	M_8	M_6	M_3	M_5	M_1	M_7	M_3	M_6
Tool	T_2	T_7	T_9	T_6	T_8	T_3	T_2	T_2	T_5	T_5	T_2	T_6	T_3	T_1	T_1
Processing time	11.4	9.69	10.20	18.10	17.52	10.83	4.51	14.43	8.20	9.70	10.45	13.90	9.58	7.25	9.25

Table 5.8 Job order of five-parts using real-time information

Problem instance 1															
Operation sequence	O_{13}	O_9	O_1	O_7	O_2	O_4	O_{10}	O_{11}	O_{14}	O_6	O_{15}	O_5	O_8	O_{12}	O_3
Machine	M_6	M_1	M_2	M_4	M_3	M_4	M_3	M_5	M_7	M_1	M_7	M_2	M_2	M_1	M_8
Tool	T_5	T_7	T_2	T_9	T_4	T_2	T_9	T_4	T_8	T_2	T_1	T_1	T_3	T_6	T_2
Processing time	5.01	3.43	4.16	4.53	4.04	3.14	3.02	3.23	6.19	5.18	4.10	4.87	4.96	5.43	7.58
Problem instance 2															
Operation sequence	O_1	O_4	O_9	O_2	O_{10}	O_7	O_{13}	O_5	O_{14}	O_3	O_{11}	O_{15}	O_6	O_8	O_{12}
Machine	M_2	M_4	M_8	M_3	M_7	M_4	M_6	M_2	M_8	M_2	M_3	M_8	M_2	M_4	M_5
Tool	T_2	T_2	T_4	T_6	T_9	T_9	T_6	T_2	T_8	T_3	T_5	T_1	T_2	T_1	T_6
Processing time	6.85	6.92	6.83	9.87	8.38	8.76	5.25	9.99	4.66	5.16	9.82	5.76	7.99	7.65	7.08
Problem instance 3															
Operation sequence	O_7	O_{13}	O_1	O_9	O_2	O_{14}	O_{15}	O_{10}	O_4	O_6	O_8	O_3	O_{11}	O_5	O_{12}
Machine	M_3	M_7	M_2	M_1	M_3	M_7	M_7	M_2	M_4	M_5	M_4	M_8	M_3	M_5	M_2
Tool	T_3	T_6	T_2	T_7	T_6	T_7	T_1	T_8	T_9	T_2	T_3	T_2	T_4	T_1	T_6
Processing time	6.3	5.90	7.68	5.73	7.17	4.41	3.62	9.98	5.58	9.61	6.56	6.43	8.43	7.75	6.52
Problem instance 4															
Operation sequence	O_9	O_{13}	O_{14}	O_{10}	O_4	O_6	O_7	O_5	O_1	O_8	O_{15}	O_{11}	O_2	O_3	O_{12}
Machine	M_5	M_7	M_7	M_2	M_3	M_1	M_5	M_3	M_3	M_4	M_8	M_5	M_2	M_1	M_5
Tool	T_4	T_6	T_7	T_9	T_2	T_2	T_9	T_2	T_2	T_3	T_5	T_4	T_6	T_3	T_6
Processing time	13.5	6.8	9.91	7.81	5.08	11.43	10.77	6.45	8.00	11.16	7.37	10.74	9.24	5.99	9.88
Problem instance 5															
Operation sequence	O_{13}	O_9	O_1	O_{10}	O_4	O_6	O_7	O_2	O_{11}	O_{14}	O_3	O_5	O_8	O_{15}	O_{12}
Machine	M_7	M_6	M_1	M_7	M_4	M_5	M_4	M_2	M_5	M_8	M_1	M_2	M_2	M_6	M_5
Tool	T_6	T_7	T_2	T_8	T_9	T_2	T_9	T_6	T_5	T_7	T_2	T_1	T_3	T_3	T_6
Processing time	8.11	14.86	10.89	8.1	5.29	11.08	11.49	7.57	9.46	8.17	10.30	6.20	11.35	6.54	7.66

An example is illustrated, to explain the functioning of agents used in the proposed WSC framework. In brief, firm A receives a manufacturing order of five parts where numbers of part types to be produced are 11, 7, 15, 15 and 10 respectively.

Step 1: New job-order request for TMA (ID 001), (Parts = 5, part types are 11, 7,15,15,10 respectively)

Step 2: Total number of defined operations = 15

Step 3: Decomposition of task –
Sub-task (a) = 1, 2, 4, 5, 6, 7, 8, 11, 12
Sub-task (b) = 3, 9,10

Sub-task (c) = 13, 14, 15

Step 4: Transmission of request message; the TMA (ID 001) request to collaborating TMA (ID 002) for operations 3,9,10,13,14,15 and TMA request to PCA for operations 1, 2, 4, 5, 6, 7, 8, 11, 12, 3, 9, 10

Step 5: Response (Table 5.9) transmission from associated PCA (ID 011) to the TMA (ID 001)

Step 6: Response (Table 5.10) transmission from Outsource TMAs (ID 002) to the TMA (ID 001)

Step 7: Generation of optimized schedule plan (Table 5.11)

Step 8: Message transmission from host TMA (ID 001) to outsource TMA (ID 002) to assign the allocated outsource task

Step 9: Transmission of defined schedule plan from TMA (ID 001) to the PCA (ID 011)

Step 10: The host PCA (ID 011) communicates with the associated MAs to allocate tasks (operations 1, 2, 4, 5, 6, 7, 8, 10, 11, 12) as per the defined operation schedules. Moreover, it communicates with the outsource PCAs (ID 012) to deal with the physical material flow and transportation aspects of the operations assigned at outsourcing MAs.

Table 5.9 Information transferred from PCA (firm A) to TMA (firm A)

Subtask-type	Operation	Real-time available alternative machines	Real-time available alternative tools	Starting time for the availability of machine
a	O_1	M_1, M_2, M_3	T_1, T_2, T_3	
a	O_2	M_2, M_3	T_4, T_6	
b	O_3	M_1, M_2	T_1, T_2, T_3	
a	O_4	M_3, M_4	T_8, T_2, T_9	M_1 = 2.0 hour
a	O_5	M_2, M_3, M_5	T_1, T_2, T_3	M_2 = 2.5 hour
a	O_6	M_1, M_2, M_5	T_1, T_2	M_3 = 1.5 hour
a	O_7	M_5, M_3, M_4	T_9, T_3	M_4 = 3.5 hour
a	O_8	M_2, M_1, M_4	T_1, T_3	M_5 = 2.0 hour
b	O_9	M_1, M_5	T_7, T_4	
b	O_{10}	M_2, M_3	T_9, T_8	
a	O_{11}	M_4, M_3, M_5	T_5, T_4	
a	O_{12}	M_2, M_1, M_5	T_7, T_6	

a: Operation performed within the firm

b: Operation that can be performed within the firm or outsourced

Table 5.10 Information transferred from TMA of firm B to firm A

Sub-task type	Operation	Real-time available alternative machines	Real-time available alternative tools	Starting time for the availability of machine
b	O_3	M_6, M_7, M_8	T_1, T_2, T_3	$M_6 = 4.0$ hour $M_7 = 1.5$ hour $M_8 = 0.0$ hour
b	O_9	M_6, M_7, M_8	T_7, T_4	
b	O_{10}	M_6, M_7, M_8	T_9, T_8	
c	O_{13}	M_6, M_7, M_8	T_4, T_5, T_6	
c	O_{14}	M_7, M_8	T_3, T_8, T_7	
c	O_{15}	M_6, M_7, M_8	T_3, T_5 , T_1	

b: Operation that can be performed within firm or outsourced
c: Operation to be outsourced only

Table 5.11 Operation sequence and corresponding machine and tools for the problem instance

Operation	O_9	O_{13}	O_{14}	O_{10}	O_4	O_6	O_7	O_5	O_1	O_8	O_{15}	O_{11}	O_2	O_3	O_{12}
Machine	M_5	M_7	M_7	M_2	M_3	M_1	M_5	M_3	M_3	M_4	M_8	M_5	M_2	M_1	M_5
Tool	T_4	T_6	T_7	T_9	T_2	T_2	T_9	T_2	T_2	T_3	T_5	T_4	T_6	T_3	T_6

5.6 Conclusions

The proposed framework facilitates an information-sharing platform for similar types of firms, which can be coupled with each other to share the operation-level information. With the real-time information sharing, the production controlling system (MAS) of an individual firm settles their planning and scheduling issues by effectively utilizing available outsourcing opportunities. At the arrival of a new job order, the host firm assigns a few non-expertise operations at the suitable outsourcing locations in order to minimize the makespan and machining cost of the production order. The firm assigns the task to the collaborating group members without interfering with their schedule plan and utilizes the idle time of the available machine at the outsource locations. Thus, with the help of information sharing, firms can cooperate with their partners by providing their expertise service support. A wide number of computational experiments are conducted to test the efficiency of the proposed framework. The results are obtained by implementing the SSA algorithm and thereby comparisons are made over different problem instances.

The proposed model of WSC can further be extended by considering the multiple objectives like cost, quality, lead-time, etc. Finally, realization of the proposed framework is still a challenge ahead.

References

Chan A, Cao J, Chan CK (2005) WEBGOP: Collaborative web services based on graph-oriented programming. IEEE Trans. Syst. Man. Cyber. part A: Syst. Hum., 35(6):811–830

Chung D, Lee K, Shin K et al. (2000) An algorithm for job shop scheduling problem with due date constraints considering operation subcontract. In: Proceeding of the 25th ICPR International Conference

Gologlu C (2004) A constraint-based operation sequencing for a knowledge-based process planning. J. Intell. Manuf., 15:463–470

Huang GQ, Huang J, Mak KL (2000) Agent–based workflow management in collaborative product development on the internet. Comput. Aided Des., 32:133–144

Hung M, Cheng F, Yeh S (2005) Development of web-service-based e-diagnostics framework for semiconductor manufacturing industry. IEEE Trans. Sem. Manuf., 18(1):122–135

Innocenti A Labory S (2004) Outsourcing and information management. A comparative analysis of France, Italy and Japan in both small and large firms. Eur. J. Comp. Eco., 1(1):107–125

Kempenaers J, Pinte J, Detand J et al. (1996) A collaborative process planning and scheduling system. Adv. Eng. Soft., 25:3–8

Khoshnevis B (2004) A linearized polynomial mixed interger programming model for the integration process planning and scheduling. J. Intell. Manuf.,15:593–605

Kirkpatrick S, Gelatt CD Jr, Vecchi MP (1983) Optimization by simulated annealing. Science, 220(4598):671–680

Kolisch R (2000) Integrated assembly and fabrication for made-to-order production. Int. J. Prod. Econ., 65:289–306

Lee DH, Kiritsis D, Xirouchakis P (2001) Optimal operation planning and sequencing: minimization of tool changeovers. Int. J. Prod. Res., 39:1649–1669

Lee YH, Jeong CS, Moon C (2002) Advanced planning and scheduling with outsourcing in manufacturing supply chain. Comput. Ind. Eng. ,43:351–374

Li L, Fuh JYH, Zhang YF et al. (2002) Application of genetic algorithm to computer-aided process planning in distributed manufacturing systems. In: Proceedings of the international conference on manufacturing automation: rapid response solutions to product development, Hong Kong, pp. 281–290

Lux A, Steiner D (1995) Understanding cooperation: An agent's perspective. In: Proceedings of the 1st international conference on multi-agent systems, pp. 261–268

Mamalis AG, Malagardis I, Kambouris K (1996) On-line integration of a process planning module with production scheduling. Int. J. Adv. Manuf. Technol., 12:330–338

Nahm YE, Ishikawa H (2005) A hybrid multi-agent system architecture for enterprise integration using computer networks. Int. J. Robot. Comp-Integr. Manuf., 21(3):217–234

Norman MS, David WH, Dag K et al. (1999) MASCOT: An agent-based architecture for coordinated mixed-initiative supply chain planning and scheduling. In: Proceedings of the 3rd International conference on autonomous agents (Agents '99), Seattle WA, USA

Petrie CJ (1996) Agent-based engineering, the web and intelligence. IEEE Expert, Dec: 24–29.

Shen W, Noorie D H (1999) Agent-based architecture for intelligent manufacturing: A state-of-the-art survey, Know. Info. Syst. Int. J., 1(2):129–156

Shukla SK, Son YJ, Tiwari MK (2007) Fuzzy-based adaptive sample-sort simulated annealing for resource-constrained project scheduling. Int. J. Adv. Manuf. Technol., 36: 982–995

Sikora R, Shaw MJ (1997) Coordination mechanisms for multi-agent manufacturing systems: Applications to integrated manufacturing scheduling. IEEE Trans. Eng. Mgmt., 44:175–187

SOAP specification [Online]. Available: http:// www.w3c.org/tr/soap

Sun Java Web site, http://java.sun.com/

Ta-Ping L, Yuehwern Y (2001) An agent–based production control framework for multiple-line collaborative manufacturing. Int. J. Prod. Res., 39(10): 2155–2176

Thompson DR, Bilbro GL (2005) Sample-sort simulated annealing. IEEE Trans. Syst. Man Cyber. part B: Cybernetics, 35(3):625–632

Tripathi AK, Tiwari MK, Chan FTS (2005) Multi-agent-based approach to solve part selection and task allocation problem in flexible manufacturing systems. Int. J. Prod. Res., 43(7): 1313–1335

Zijm WHM, (1995) The integration of process planning and shop-floor scheduling in small batch part manufacturing. Ann. CIRP, 44:429–432

Chapter 6
Isoarchic and Multi-criteria Control of Supply Chain Network

F. Ounnar and P. Pujo

Abstract Supply chains and more particularly supply chain networks are more and more subjected to extreme dynamic operations, where it is asked that each actor has more flexibility and reactivity on the one hand and a specialization bringing more productivity on the other hand. Companies try to achieve the common goal of satisfying customers' needs through partnership. Negotiation between partners is thus required involving each partner management and production organization. This situation makes it difficult to obtain the best response with respect to the need of each customer. For that, a new approach is proposed for customer–supplier relationship control, in which the partnership is considered in the context of an association of potential suppliers within a network: an isoarchic control model for a supply chain network based on a holonic architecture. The decision-making mechanism is produced thanks to the properties of a decision-making center, called autonomous control entity, associated to each actor of the logistic network, which makes it possible to quantify a multi-criteria evaluation. An implementation of the simulation of such a system is done via a distributed simulation environment high-level architecture). A case study is presented.

Keywords Holonic, multi-criteria, supply chain network, isoarchic control model

6.1 Introduction

Supply chains and more particularly supply chain networks are more and more subjected to extreme dynamic operations, where it is asked that each actor has more flexibility and reactivity on the one hand and a specialization bringing more productivity on the other hand. This forced many companies to search for new forms of organizations. Studies have been conducted on the durability of customer supplier relationships (Alcouffe and Corrégé, 1999), the dynamics of these relationships and their influence on inter-company costs (Brandolese *et al.*, 2000,

F. Ounnar (✉) and P. Pujo
LSIS, UMR CNRS 6168, Aix Marseille University, France
e-mail: fouzia.ounnar@lsis.org

Harri, 2002) with the aim to improve supply chain management productivity and effectiveness. In addition to these, we can mention the work of Toolea and Donaldson (2002) on customer–supplier (C-S) relationship performance and of Nesheim (2001), Smart and Harrison (2003) and Holmlund-Rytkönen and Strandvik (2005) on the impact of bidding within C-S relationships. Other studies focused on the definition of concepts allowing improved cooperation between companies (Telle *et al.*, 2004, Lauras *et al.*, 2003) or proposed an autonomous decentralized optimization system based on material requirement planning, such as the work of Nishi *et al.* (2005).

Today, the customer is placed at the centre of the organization. So, companies try to achieve the common goal of satisfying customers' needs through partnership. Partnership control involves all the actions developed together in order to achieve common objectives and to timely react to any failure of any partner. Negotiation between partners is thus required involving each partner management and production organization. This situation makes it difficult to obtain the best response with respect to the need of each customer.

For that, C-S relationship control at a tactical/operational level is proposed in which the partnership[1] is considered in the context of an association of potential suppliers within a network and in which all C-S partners negotiate according to a contract-net-type protocol (Smith, 1980) in order to meet customer requirements as much as possible. The goal of this approach is to provide support for controlling a meshed logistics network through the dynamic behavior of C-S relationships at network level. By meshed logistics network is meant a supply organization in which several supply solutions are offered at each step (i.e., at each mesh of the chain). The difficult part of such an organization is to select, for each supply step, the best solution among all the possible ones. The proposed approach is an isoarchic and multi-criteria control model for supply chain networks based on a holonic architecture. Indeed, the isoarchic approach can be implemented via the holonic paradigm, given specific software developments. Each supplier organizes and controls his own activities, obtained by proposing his best conditions for the execution of calls for proposals (CFPs) launched by customers. The decision-making mechanism is produced thanks to the properties of a decision-making centre, called an autonomous control entity (ACE), associated to each actor of the logistic network, which makes it possible to quantify a multi-criteria evaluation.

After introducing the holonic paradigm and self-organized control in C-S relationships (Sectis. 6.2 and 6.3), a two-step supplier self-assessment approach, with respect to received CFP, is described in Sect. 6.3. In order to validate the proposed approach, it is necessary to validate first the global partnership network model. For that, a distributed simulation platform using high-level architecture (HLA) standard (IEEE P1516)) is developed in Java (Sect. 6.4). Validation is done with a study case from the cosmetic industry by comparing operations under classical

[1] The goal of the partnership is to collectively ensure the dispatching of orders from different customers, while respecting each partner's interest.

control with the proposed self-organized operations (Sect. 6.5). Sect. 6.6 concludes the chapter and presents future direction works.

6.2 Supply Chain Management Limits

The bullwhip effect is observed in forecast-driven distribution channels and more generally in supply chain processes (Agrawal *et al.*, 2009). This phenomenon has different causes related to behavioral or operational problems. Behavioral causes depend on base-stock policies, erroneous feedback and time delay data, overreactions after non-forecasted demand and on poor risk assessment. Operational causes depend on demand forecasting, lead time variability, lot-sizing/order synchronization difficulties, forward buying and trade promotion or shortage anticipation. Logistics chain vulnerability is essentially due to their forecasting needs and to unavoidable related lack of precision.

Actual logistics networks are made from a set of frozen logistics chains, in the sense that each customer has his own suppliers to whom he places orders. Also, suppliers know their customers and have advance knowledge of orders. C-S relationships are defined according to a precise commercial contract with specific commitments on both sides (customer or supplier) on product quantities and/or delivery schedule. Thus, customers are constrained by suppliers and suppliers are constrained by customers. We refer to this type of logistics chain as a "static logistics chain". In such a context, relationship degradation can occur due to several problems, for instance in relation to a supplier's reaction when additional products are ordered or to a customer's reaction in the case of delayed delivery.

Problems in actual logistics networks have an impact on the fragility of network partner relationships, as a consequence of opportunist behavior of most of the partners. This led us to propose logistics chain organization within a network. That is, each actor is a node of a meshed network in which links represent potential relationships between one customer and his suppliers and in which logistics flows are dynamic and re-negotiated in each instance. Each partner of the network may belong to one or several logistics chains, according to his actual work load. Partners contribute to a common objective, which is to collectively ensure the dispatching of orders coming from various customers while respecting each customer and supplier interest. The partners are thus part of a self-organized logistics network, which we define as "a logistics network in which flow organization comes only from direct coordination and cooperation relationships between suppliers".

Contrary to actual logistics networks in which chains are frozen, a chain of the proposed self-organized logistics network is built each time an order is to be achieved; nothing is planned ahead of time. Meshing allows defining several paths to implement the supply chain; the possibility of several potential paths allows avoiding supply shortage that can be seen in a supply chain when a mesh of the chain is deficient. Of course, the proposed approach is efficient if control decisions are performed in real time without forecasting. This can be based only on a strong reactivity of the logistics network actors who must put in common and share knowledge. We define a "dynamic logistics chain" as a logistics chain progressively built within the logistics network.

An approach based on the holonic paradigm and isoarchic architecture for self-organized C-S relationships is proposed. In this approach, all the partner entities (customers, suppliers) of a self-organized logistics network exchange through the same communication media and negotiate to answer at best customers' expectations and to exploit at best suppliers' capabilities. In the proposed self-organized network, customers launch CFPs, potential suppliers enter negotiations from which the best network answer emerges for each CFP and allows identifying the next mesh of the logistics chain.

The logistics network is thus built on honest and transparent partnership. An important condition for the good operation of this type of relationship is the existence of mutual trust among partners. It is necessary to sensitize and engage logistics people into a policy of permanent progress, made of continuous improvement with the aim to maximize economic potential. Beyond internal optimization of their production, and in connection with it, customers must optimize their relationships with their suppliers. Each supplier positions itself with respect to the various customers and shows its capacity to provide the needed support while letting each partner use its own assets.

A durable C-S relationship in a dynamic partnership requires the use of an appropriate set of tools supporting items such as contractual relationships, trust development between partners and assessment of relevant, coherent and motivated suppliers. Trust, reciprocity and shared goals are the principal components of a strong C-S relationship.

6.3 Control of a Dynamic Logistic Network: Isoarchic and Multi-criteria Control

A self-organized control model is proposed in which the decision system manages the operations of a set of entities belonging to a partnership. The self-organization concept is conditioned on one hand to the use of a decentralized decision structure and, on the other hand, to the consideration of the real behavior of each entity. With this approach, there is no forecast-based organization since self-organization implies a real-time decision-making mode. For that, and also to allow each supplier to take part in negotiations, an ACE has been associated to each partner. This entity allows dialog with the other network members and self-assessment with respect to received CFPs.

The main objective of our approach is to ensure the dispatching of customers' orders through negotiations between the suppliers potentially able to answer a CFPs issued by a customer. Because of the limited capacity of each supplier, it is important to respect a load/capacity balance at suppliers' level. This leads to automatic load smoothing between potential suppliers, which yield, on a longer term, a fair load distribution system between the network suppliers.

6.3.1 Description of the Interacting Entities

The holonic paradigm is proposed by Kostler (1967) for complex social systems modeling. In such systems, a holon is at the same time a whole and part of a whole (Janus effect). This view marks a break from former hierarchical models in which behaviors are of the master–slave type according to a tree-like organization and an invariant topology of the decision-making centres, the whole being reinforced by the respect of orders by the decision-making centre slave. A holon has a decisional intelligence, which enables it to act on its own behavior and also to intervene in the behavior of the system it belongs to (Pujo and Ounnar, 2007).

The hierarchical decomposition is replaced by a recursion of holons and the implementation of the Janus effect. This provides a wide scope for control system implementation according to hierarchical architecture (Trentesaux, 2007), i.e. being able to mix centralized and non-centralized decision centres.

Several holonic architectures are proposed in the scientific literature for holonic manufacturing system control. The best known is PROSA (product, resource, order, staff, architecture) (Van Brussel *et al.*, 1998). This reference is used to derive a self-organized control system for C-S relationships based on a holonic architecture, which includes:

- Resource holons (RH), corresponding to a company of the logistics network partnership and carrying production capacity characteristics.
- Product holons (PH), providing a technical description of the manufactured products (manufacturing process data, models, sequences, etc.) and thus completing production task specification. This type of holon contains all information on product traceability. There are as many PHs as manufactured products or as products in the work in progress (WIP). This is a major difference with PROSA.
- Order holons (OH), representing a task in the production system. A work order concerns a set of PHs. The OH is closely related to the concepts of batch, WIP and lead times.

The proposed architecture respects a flat holonic form between ACEs (Bongaerts *et al.*, 2000). Thus, the main role of ACEs is to manage all information exchanges within the network and to organize the information-processing sequences leading to decision-making, without hierarchical subordination links and/or forecast calculation. Supply chain management works then in a self-organized control mode within an isoarchic architecture. The problem of self-organization is based on how product flows are built in real time. In fact, the true problem is the fate of the product being processed onto the resource, i.e. which resource will be in charge of the next stage or which product will be processed by a given resource (Pujo *et al.*, 2009).

6.3.2 Definition of Self-organized Control

Self-organized control implies decision-making in real time without forecasting (Ounnar and Pujo, 2001). In order to organize, one must define a common objective for all the decision centres involved in the organization. This can be done with different ways according to the properties and characteristics of the entities to be organized, and expressed in terms of synchronization, coordination, sharing, co-operation, negotiation and/or operation design. The solution retained to make all the entities work is obtained by emergence. Indeed, in absence of hierarchy, each entity participates in the solution proposal definition and also in the solution evaluation. The proposal that appears the best performing with respect to pre-established criteria is selected.

Many ways exist to locally handle decision-making for each partner. The use of scheduling heuristics could be envisaged at this level. A different approach is chosen aiming to maximize the consideration of, sometimes diverging, customers' and suppliers' interests. As already mentioned the objective of our approach is to make emerge the supplier answering best a CFP launched by a customer into the partnership network, while respecting the interest and exploiting at best the capacity of each partner. In a first step, a supplier sorts out all the received CFPs with the objective to define the most advantageous CFP for him. In a second step, a local assessment is performed for the CFP selected in step 1. This individual evaluation allows the supplier to position himself with respect to the best response circulating on the network for the selected CFP. This way, the selection of a supplier proposing the best performance for a given CFP will emerge and this supplier will be in an optimal situation. If a supplier is not retained for his selected CFP, he can then bid on other CFPs that he did not rank as high.

Both steps can be implemented with the use of qualitative and quantitative criteria. Three classes of multi-criteria methods can be distinguished (see Figure 6.1): decision aid methods, elementary methods and mathematical optimization methods. The choice of one class of methods may depend on the data available to treat the multi-criteria problem or on the way the decision-maker models his preferences.

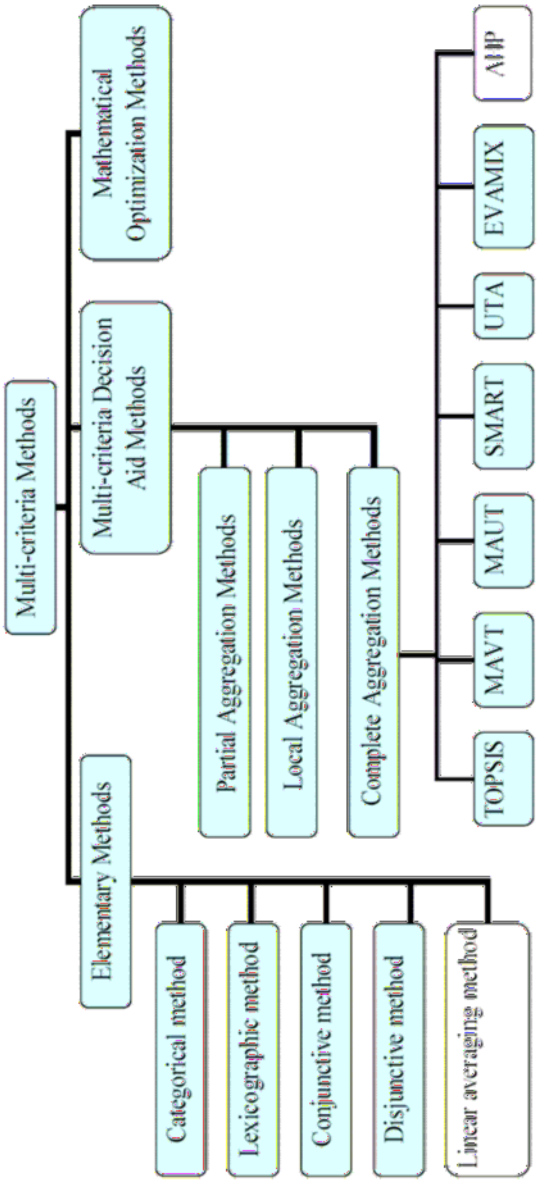


Fig. 6.1 Multi-criteria method used for each step

These aspects are described below:

1. Choice of elementary methods or multi-criteria decision aid methods. These methods are used when the problem addressed has several solutions. In this case, the problem is reduced to the selection of the best method. However, the selection of the best method is conditioned by the way the decision-maker expresses his preferences. This is what is called preferences modeling in decision theory and what differentiates the two types of methods. In multi-criteria decision aid methods, the decision-maker may not express preference between two possible solutions. In elementary methods, indifference cannot be modeled.
2. Choice of mathematical optimization methods. These methods allow handling problems with constraints for which solutions are not known. The used models can be linear, linear with integer variables, mixed linear, quadratic, non-linear, etc.

In our case, the selection process of the CFP to be "processed" by an RH (supplier) supposes knowledge of the various possible alternatives in order to carry out sorting with respect to a set of criteria. So, the use of optimization methods is not possible.

Because of their ability to integrate very different and often antagonistic criteria, decision aid methods have been developed to handle multi-objective problems (Ben Mena, 2000). These methods provide the decision-maker with useful help during the refinement of his decision process for the selection of one action among a set of potential actions or for the ranking of a set of actions through examination of the logic and coherence of his preferences. This class of methods is analyzed to select a method for CFP ranking. The second step concerning performance evaluation of the first-ranked CFP is based on the analysis of the class of elementary methods.

6.3.2.1 First Step: CFP Ranking

This is about ranking all the alternatives by examining logic and coherence of choices and then by aggregating preferences according to one of the three approaches: complete, partial or local. Preference aggregation consists in exploiting partial action evaluation on the various criteria in order to come up with a global evaluation. In our case, complete aggregation is applied for ranking all the CFPs received by the RH (supplier). Assessing a supplier's performance with respect to a selected CFP must be the results of the interaction of all the considered criteria and no criteria can be put aside during the evaluation process. There exist several methods in this context (Ounnar, 1999, Mekaouche, 2007): TOPSIS (technique for order preference by similarity to ideal solution), MAVT (multiple attribute value theory), MAUT (multiple attribute utility theory), SMART (simple multiple-attribute rating technique), UTA (utility theory additive), EVAMIX (evaluation of mixed criteria), AHP (analytic hierarchy process).

The AHP (Saaty, 1980) was chosen. It has several advantages over other decision-making approaches (Vargas, 1990, Wedley, 1990), which include ability (i) to handle qualitative and quantitative attributes, (ii) to hierarchically structure problems to gain insights into the decision-making process and (iii) to monitor the judgment consistency of a decision-maker. Furthermore, AHP has the capability to quantify and rank the alternatives (CFPs) using pairwise comparison of criteria (Harker, 1989). All these characteristics make the strong points of the AHP method (Ounnar, 1999). AHP has demonstrated robustness across a range of application domains.

A multi-criteria decision-making algorithm applying AHP is established in each ACE associated to an RH and defines a CFP classification, taking into account RH, PH and OH constraints. The performance for the CFP ranked in first position will be taken into account. We propose to implement AHP through two main phases: configuration and exploitation. In order to be able to use the AHP algorithm for ranking the received CFPs, it is first necessary to define the relative importance of the criteria and their indicators: this is the setup (configuration) phase. The dynamic exploitation phase of the AHP algorithm makes it possible to rank the CFPs and obtain the priority vector of the considered CFPs.

In our study, five criteria are proposed (Ounnar *et al.*, 2007):

1. Cost criterion C1. The objective of this criterion is to ensure delivery at the best price. This criterion is a quantitative criterion that takes into account the various costs in the acquisition of goods. The cost criterion can be reduced to two indicators: cost of order I11 and cost of order delivery I12.
2. Lead time criterion C2. The objective of this criterion is to ensure that the customer receives delivery as quickly as possible. The delay is the time between the expression of a need by the customer and the actual satisfaction of this need. This criterion can be reduced to two indicators: production time I21 and delivery time I22.
3. Quality criterion C3. The objective of this criterion is to guarantee that the delivered products are of good quality and in accordance with specifications, that is, the criterion aims to minimize poor unquality. The indicators for this criterion can be quantitative or qualitative, and aim at describing continuity of service, compliance with the rules and compliance with expectations concerning the product. This criterion can be reduced to three indicators: rate of conformity I31, respect of a referential I32 and rate of customer satisfaction I33.
4. Reliability criterion C4. Reliability is the ability of any device to carry out a required function, under given conditions, for a given duration. The objective of this criterion is to guarantee that the delivered products are reliable. This criterion can also be used to evaluate the capacity of the company to meet deadlines. This criterion can be reduced to two indicators: conformity in quantity of the orders I41 and respect for delivery times I42.
5. Strategy criterion C5. In the evaluation of each supplier's performance, qualitative criteria are taken into account. For example, privileged relationships link

the customer and supplier. This criterion can be reduced to two indicators: ailowance of differed payment I51 and degree of privilege I52.

This section defined a system of indicators for the application of the multi-criteria decision method (AHP) to the evaluation of each potential supplier. The application of the proposed multi-criteria method provides the alternative (the CFP) for which the supplier is the best.

6.3.2.2 Second Step: Performance Assessment for the First-ranked CFP

As indicated above we have focused on the class of elementary methods. These methods are often implemented in practice (Vincke, 1989). In general, the decision-maker assigns a weight P_c to each criterion representing the criterion relative importance. Then, the decision-maker associates a mark P_{ac} to each action with respect to each criterion. The global mark F_a of each action with respect to n criteria is calculated as $F_a = \sum_{c=1}^{n} (p_c)(p_{ac})$. Among elementary methods, we can mention (Timmerman, 1986): categorical method, linear averaging method or weighted point method; (Akbari Jokar, 2001): lexicographic method, conjunctive method and disjunctive method.

The selected method should authorize consideration of qualitative and quantitative criteria, rely on all (without exclusion) the chosen criteria and send only one (not several) action as result. The categorical method and linear averaging method meet these three constraints. The unique difference between these methods is related to criteria weighting. The linear averaging method is complementary to the categorical method, which is used to assess supplier's performance for a selected CFP. Indeed, the linear weighting method cannot be directly applied for three main reasons:

- The considered indicators are not necessarily of the same nature.
- The linear weighting method, as it is defined, presents a monotony problem.
- The objective is not only to weight the various criteria but also to penalize or neutralize indicators according to customer wish.

The five selected criteria used for assessment are considered as the coordinates of an element in $\Re 5$. We thus work in an $\Re$-vector space of dimension 5. With the aim to eliminate the monotony problem and to be able to penalize a supplier, a mark is given to the indicators in order to discriminate well enough, at the end, between the performances of the various suppliers on similarly chosen criteria. Performance evaluation is thus based on three main points:

1. Respect of the scales between indicators.
2. Coherence of the scale between criteria.
3. Implementation of coefficients k_1 and k_2 to be able to eventually penalize a supplier.

A supplier performance for a selected CFP (in step 1) is calculated by:

$$\left\|P_{\text{locale}}\right\| = \sqrt{\begin{array}{l} [p_1 \times k_1]^2 + [p_2 \times k_2]^2 + \left[p_3 \left(\dfrac{I_{31} + \dfrac{I_{32}}{9} + I_{33}}{3} \right) \right]^2 \\ + \left[p_4 \left(\dfrac{I_{41} + I_{42}}{2} \right) \right]^2 + \left[p_5 \left(\dfrac{\dfrac{I_{51}}{9} + \dfrac{I_{52}}{9}}{2} \right) \right]^2 \end{array}} \tag{6.1}$$

where p_1, p_2, p_3, p_4 and p_5 are constant coefficients providing customers with the possibility to privilege one criterion above the others by giving weights with total sum equal to 1.

The coefficient k_1 is defined by: (desired cost)/(possible cost) = k_1; “Desired cost” corresponds to the cost that a customer is able to accept and “Possible cost” reflects the cost proposed by the supplier to customer (I_{11}+I_{12}).

Coefficient k_1 is equal to 1 if the possible cost is lower than the cost desired by the customer. The following algorithm describes the calculation: if desired cost < possible cost, then k_1= (desired cost)/(possible cost); if not, $k_1 = 1$.

As for criterion C_1, a coefficient k_2 is introduced for criteria C_2 with the aim to penalize a “bad” supplier proportionally to the difference between his performances and customer expectations. The following algorithm allows that: If desired date < possible date (with possible date = I_{21}+I_{22}), then k_2 = (desired date) / (possible date); if not $k_2 = 1$.

6.3.3 Support Structure: Autonomous Control Entity

A control system which is at the same time decentralized, at a single decision level, and self-organized, in real time, can be characterized as isoarchic. The definition of “isoarchy” comes from the two Greek radicals: *iso*-(equal) and *archy*-(power). It thus means the same authority for the different decision-making centers and a total absence of hierarchy, in function as well as in time. In a decision system composed of several decision-making centres, a decisional architecture can be described as isoarchic when each decision-making centre is equipped with the same capacity of decision (it is a collective decision because each concerned holon brings its own decision into the decision-making process). This property can easily be obtained when the decision mechanisms are duplicated on each decision-making centre and are parameterized according to the characteristics of each of them. Isoarchy appears as a particular specification of the heterarchy concept and the opposite of the hierarchy concept (Mesarović *et al.*, 1970). The concept of isoarchy can be implemented via the holonic paradigm.

The proposed architecture for the self-organized control model is a flat holonic form, which can be seen as an isoarchical particularization of a holonic architecture. In this architecture, the basic role of the ACEs is to manage all information exchanges in the network linking the different entities and to organize information processing leading to decision-making. Each ACE is equipped with a set of services aiming to organize the interactions between holons, so as to support productivity according to criteria that can be parameterized. Other services like archiving, traceability or performance assessment can be hosted by an ACE. ACEs allow entities to self-assess themselves against received CFPs. ACEs are in fact at the heart of the relationships between the base holons. An ith CFP (CFP*i*) is received by the ACEs. Each CFP is composed of information coming from the PH and the OH (see Figure 6.2). The performance value (Perf) is sent on the network via the response to the RCFP*i*. To obtain this performance, the optimization module needs operating time data, which are obtained from the resource holon (the company). Each ACE has privileged exchanges within the resource holon it is associated with, which provides information on its planning, its capability, etc. The data associated with OHs and PHs circulate in the network via CFPs. Products are associated with PHs describing them through their progress along the production system, according to their routing. Their progress is managed by the evolution of the OHs that trigger production tasks.

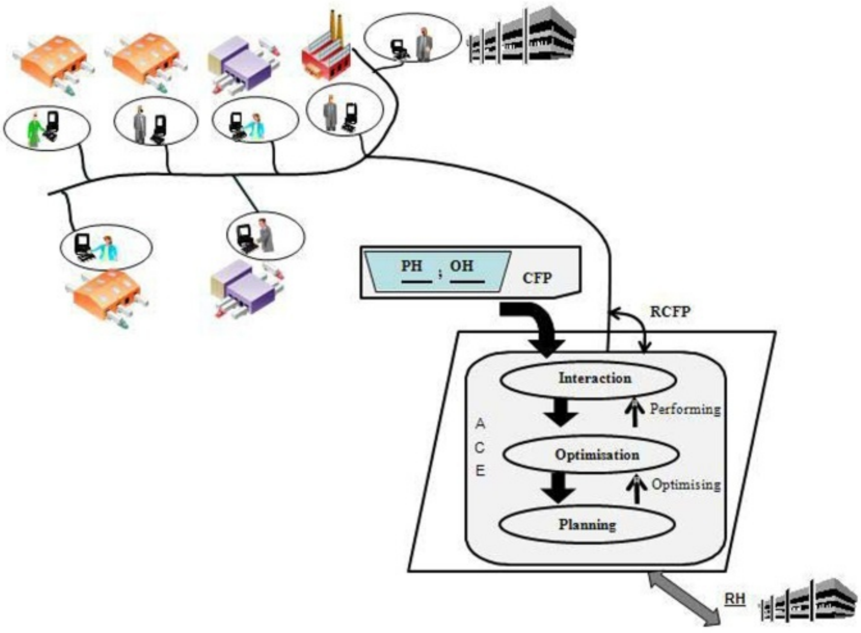


Fig. 6.2 Relation among partners via the flat holonic form

The isoarchic character of control is based on the interaction, optimization and planning modules of the ACE. The interaction module allows CFP assignment by calling for competition between the RHs (suppliers) that are potentially able to answer the CFP. In order to obtain a coherent result, it is necessary to implement a decision protocol based on impartial and common rules and criteria. This decision mechanism belongs to the interaction protocol family of the contract net family (Smith, 1980, Hsieh, 2008). It consists in determining within a network of entities the one that gives the best response to a CFP. The interaction module also manages the status of the various CFPs being processed, according to the evolution of task assignment and to the response to CFPs circulating on the network:

- "Negotiable" CFP: a CFP being negotiated.
- "Committable" CFP: a CFP temporarily assigned to a RH because its performance is the best one at that moment; this status can be lost if a better offer (response) appears and regained if, for example, new possibilities appear on the RH planning.
- "Pre-committed" CFP: a "Committable" CFP being the next one to be processed on the RH planning (the negotiation completion date has been reached).
- "Committed" CFP: a CFP that is pre-committed and whose implementation is on-going or about to start, which makes it definitively placed on the resource planning at the commitment date of this entity.

The model of this interaction protocol is described according to the DEVS (discrete vents system specification) formalism in Ounnar *et al.* (2009). The emergence of the best solution will occur under equity conditions supporting the search for the best solution.

The optimization module allows the ACE to self-evaluate the capacity of the resource to carry out the task described through the received CFP: the ACE estimates its ability and its own performance to answer. Performance evaluation is based on the fact that all the ACEs in competition for a CFP use an identical calculation mode to assess their performance; what makes the difference between results are differences in production system characteristics: current state at the time of the calculation and WIP status. For that, it is necessary to use the planning module.

The Planning module allows building the production planning of the RH. It is not definitive planning, but very short-term dynamic planning. The resource planning contains in chronological order: the task under execution, which is committed, then the following task, which holons have firmly committed themselves to perform and which is pre-committed. The rest of the planning comprises a set of pre-committed tasks and committable tasks. The latter are temporarily positioned on the planning, waiting for their commitment or their suppression. Responses to CFPs are evaluated according to this planning, while trying to find a place for the negotiable tasks.

6.4 Distributed Simulation of a Dynamic Logistic Network

Validation of the proposed approach goes through the validation of the global model of the partnership network. For that, each ACE is modeled according to the DEVS formalism (Ounnar *et al.*, 2009). With the aim to respect partners' data confidentiality, all these models have been integrated into a distributed simulation via the HLA standard. This allows, through a simulation mock-up dedicated to the proposed approach, performing a formal analysis of synchronization problems. HLA has also the capability to implement interoperable simulations with real operations.

6.4.1 High-level Architecture Components

The HLA developed by the Defense Modeling and Simulation Office (DMSO) of the Department of Defense (DoD) defines an approach to integrate federates (components of simulation) in one distributed simulation system. It facilitates reusability and interoperability of simulations. Reusability means that simulation components models can be reused in various simulation applications without need of recoding. Interoperability implies the capacity to combine simulation components on distributed platforms of various types. HLA federation means a distributed simulation system using a set of elementary simulations exchanging information called federates. HLA is formally defined by three components:

1. HLA rules: the principles of HLA are summarized in 10 rules defining the federate operation (5 rules) and federations (5 rules) (IEEE P1516).
2. HLA interface specifications: they describe the execution services provided to a federate by the run-time infrastructure (RTI). The RTI manages communication among the different federates, the organization and the distribution of the exchanged messages within a global simulation. The HLA interface specifications indicate how federates interact during the federation execution (IEEE P1516.1).
3. OMT (object model template) is the description of the elements (objects and interactions) that are shared through a federation (IEEE P1516.2). HLA requires that each federate and the federation documents their object model by using the OMT. These models are designed to facilitate the reusability of simulations.

HLA specifies two types of objects models. The first one is the federation object model (FOM), which describes the set of objects, attributes and interactions that are shared through the federation. The second one is the simulation object model (SOM), which provides information on simulation capability for information exchange taking part in a federation.

6.4.2 Integration of the DEVS-ACE Models in High-level Architecture Environment

With the aim of integrating the DEVS-ACE models in HLA environment, the ACE federate is split up into (see Figure 6.3):

- Simulators of the DEVS-ACE models: they ensure the simulation of the atomic models by using the functions defined in DEVS to generate their behavior.
- Local Coordinator: it insures the routing of the messages among the various simulators of atomic DEVS models according to coupling relations.
- FOM: it contains the various tables, which are necessary to define the OMT. For example, we find in the FOM the interaction class structure table. The interactions classes are needed to establish communication among federates, they move the messages containing the output events influencing other federates. It is thus necessary to translate the DEVS messages by using the HLA interaction classes.
- Interface among model simulators and the RTI (simulation kernel): The RTI manages communication, FOM data exchange among the different federates, recovery, organization and distribution of the messages exchanged within a global simulation. The RTI also plays the difficult role of ensuring temporal synchronization among federates while respecting the principle of causality (the order of treatment of events). The role of the simulation kernel is to manage the simulation.

6.5 Experiments via Simulation

The work described above is implemented in an application by using the Java language. An ACE is implemented which can be duplicated for all the partners of the network. Let us consider a network composed of one customer and four suppliers. A partner can become a member of the partnership network through the interface (see Figure 6.4). This partner can be a customer and/or a supplier. The customer has broadcasted a CFP on the network containing data obtained from PH and OH: product reference (PH), type of work to be performed (in technical terms; PH), theoretical time to perform the task (here 55 min; PH), maximum expected lead time (here, 10 days; OH), batch quantity (here 22 parts; OH), dead line for end of negotiation (here 105 min, OH), … CFP broadcasting is done thanks to HLA interactions by using the methods of the RTI ambassador. The simulation kernel translates the received CFP into a DEVS event and sends it towards the corresponding simulator of the atomic model via the local coordinator.

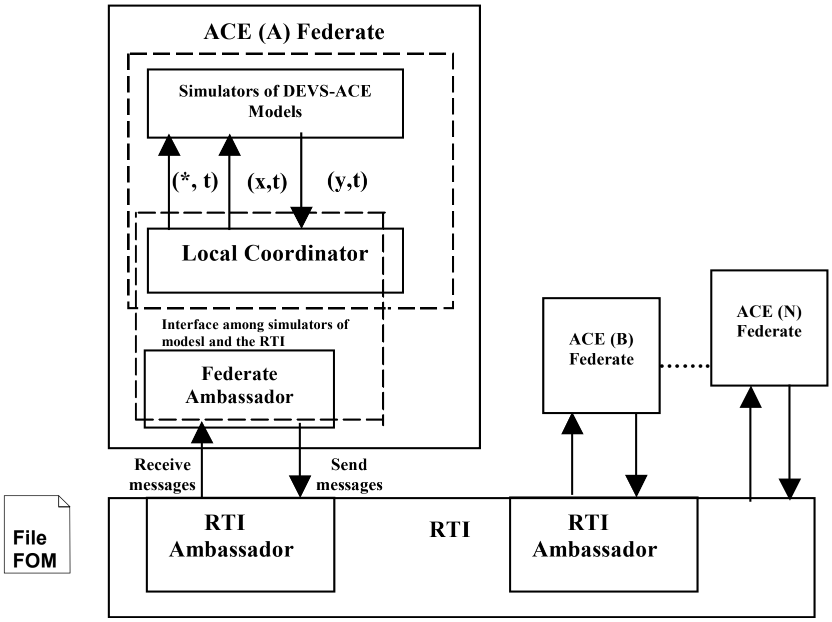
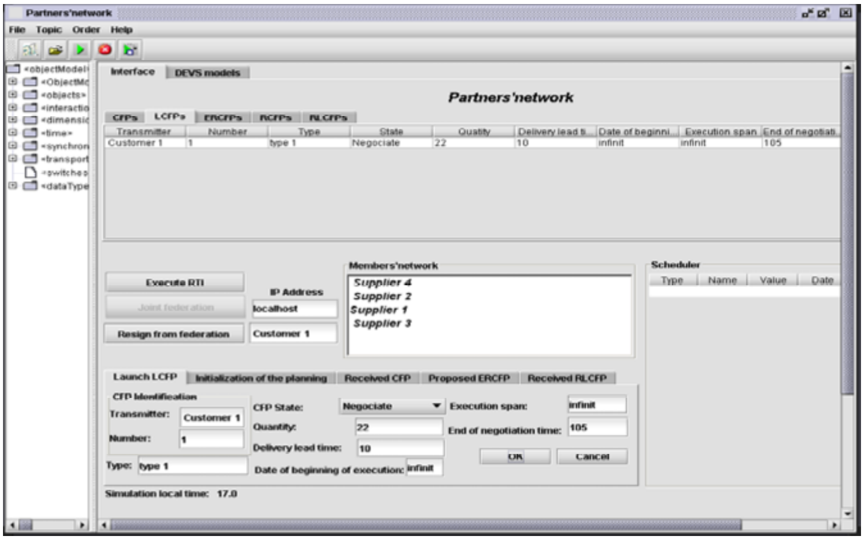


Fig. 6.3 Integration of the ACE-DEVS models in HLA environment



Partners'network

Transmitter	Number	Type	State	Quatity	Delivery lead ti...	Date of beginni...	Execution span	End of negotiati...
Customer 1	1	type 1	Negociate	22	10	infinit	infinit	105

IP Address: localhost
Customer 1

Members'network
Supplier 4
Supplier 2
Supplier 1
Supplier 3

Scheduler

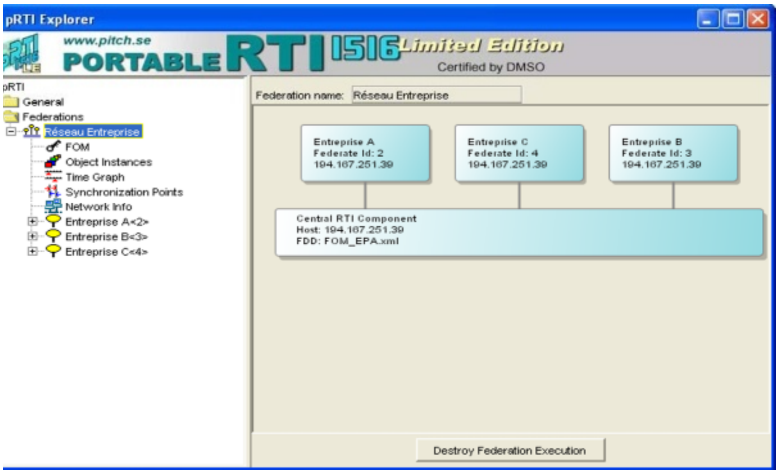
Type	Name	Value	Date

CFP Identification
Transmitter: Customer 1
Number: 1
Type: type 1
CFP State: Negociate
Quantity: 22
Delivery lead time: 10
Date of beginning of execution: infinit
Execution span: infinit
End of negotiation time: 105

Simulation local time: 17.0

Fig. 6.4 ACE federate interface

Exchanges between federates are made possible through the HLA federation (see Figure 6.5).



Federation name: Réseau Entreprise

Entreprise A
Federate Id: 2
194.167.251.39

Entreprise C
Federate Id: 4
194.167.251.39

Entreprise B
Federate Id: 3
194.167.251.39

Central RTI Component
Host: 194.167.251.39
FDD: FOM_EPA.xml

Fig. 6.5 Federation of companies

Validation of the proposed self-organized control is done through a case study by comparing operations under classical "static logistics chain" control and operations under "dynamic logistics chain" self-organized control.

The case study comes from cosmetic industry. A network made of 17 enterprises is considered involved in six main activities (cosmetic product design and production, paper packaging of products, plastic product fabrication and plastic conditioning for products, printing, etc.) and able to perform other secondary activities. Sixteen types of products coming from five main finished products (PF1, PF2, PF3, PF4 and PF5) are manufactured at different levels by these enterprises. Each enterprise may be a customer or a supplier, depending on the type of product considered. For the five main finished products, the various types have the same bill of materials and the same routing; they do not have the same unit production time and do not necessarily go through the same enterprise chains. There are 16 logistics chains corresponding to the flows of the 16 different product types. The flows of the 16 product types and the associated chains are studied.

The set of data is adjusted so that small perturbations of nominal operations (like increase of the quantities ordered by the customer to his supplier) create strong organizational perturbations in flow progress (blockage, saturation). In a first step, networked enterprise operation in the context of static logistics chain control (constant flows and quantities) is studied by modeling and simulating a set of relationships between companies. In a second step, the same network (same production capacity, same processed products) is studied in the context of dynamic logistics chain control implemented on the self-organized logistics network mock-up under the HLA simulation environment.

The influence of different types of perturbations introduced in the two approaches is analyzed (Ounnar *et al.*, 2009): perturbation on increased number of ordered products at the level of the busiest enterprises and at the level of enterprises involved in the manufacturing of several products, and perturbations on breakdown handling. The objective being to show how supplier enterprises react in the case of a classical approach against situations in which customers' expectations are not the same as expected.

It can be seen in Figure 6.6 that for each enterprise, the Occupation Rate (OR) shows frequent overloading in the classical approach and does not reach saturation level in the self-organized mode for the same activity volume. There is load smoothing and load balance. Enterprises whose work load exceeds their maximum capacity in the classical approach do not address calls that would create overloading in the self-organized approach. These calls are handled by other enterprises and the possible perturbations (delay, partial delivery, etc.) can no longer feed the bullwhip effect.

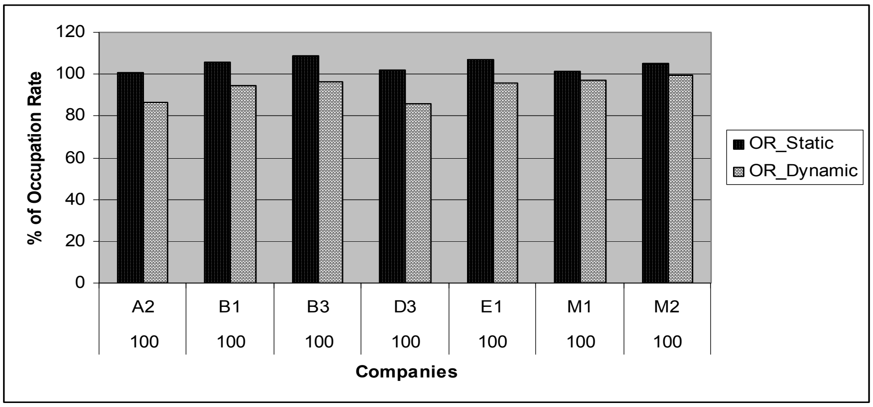


Fig. 6.6 Occupation rate of A2, B1, B3, D3, E1, M1 and M2 companies

6.6 Conclusion and Future Work

An approach for self-organized control of a logistics network is presented based on the holonic paradigm and isoarchic architecture. The objective is to collectively ensure the dispatching of orders coming from different customers, while respecting everyone's interest. When a customer launches a CFP, potential suppliers enter a negotiation phase and the best response emerges according to suppliers' actual status. This type of network is based upon partnership. It is a logistics network in which logistics people are engaged into a sustainable progress approach made of continuous improvement aiming to maximize economic potential. For customer companies this leads, beyond their own internal optimization and in relation to it, to optimize their relationships with suppliers. This allows each supplier to position themselves with respect to various customers and to demonstrate their capability to be integrated into a support need, while allowing everyone to play his own assets (Ounnar and Pujo, 2001). Data sharing at the various stages of the network will feed common effort. Logistics networks will feature flexible scheduling and delivery dates will be respected for *ad hoc* products. Deliveries circuits will be set with enhanced precision and flexibility without additional cost. Information systems will be generalized and the network members will work in a cooperative environment, within a global strategy, instead of seeking local benefits to the disadvantage of other network actors.

The proposed approach provides a balance between load and capacity for the supplier company and produces a load curve smoothing among the network suppliers with the long-term objective of establishing a fair system among them. It also leads to earnings sharing, resources optimization, malfunctioning reduction, and productivity increase for the whole supply chain. This approach allows data confidentiality for the different network partners. Indeed, a supplier company provides only a single piece of information to the network, which is his performance value for a given CFP. Consequently, the approach can be applied within a logistics network in which there is mutual trust among the partners or in a multi-sited company.

The proposed system can be deployed for an industrial case. The capability of HLA to perform simulations in parallel to real operations will make comparisons and validation possible. Results analysis will allow pointing out situations for which the proposed approach yields significant results.

References

Agrawal S, Sengupta RN, Shanker K (2009) Impact of information sharing and lead time on bullwhip effect and on-hand inventory. Eur. J. Oper. Res., 192(2):576–593

Akbari Jokar MR (2001) Sur la conception d’une chaîne logistique: Une approche globale d’aide à la décision. PhD, Institut National Polytechnique de Grenoble – France

Alcouffe C, Corrégé N (1999) L’évaluation des performances dans les organisations en réseaux de sous-traitants: l’exemple de l’Aérospatiale Matra Airbus. RFGI, 18(4):27–42

Ben Mena S (2000) Introduction aux méthodes multicritère d'aide à la décision. Biotechnol. Agron. Soc. Environ., 4(2):83–93

Bongaerts L, Monostori L, McFarlane D et al. (2000) Hierarchy in distributed shop floor control. Comput. Ind., 43:123–137

Brandolese A, Brun A, Portioli-Staudacher A (2000) A Multi-agent approach for the capacity allocation problem. Int. J. Prod. Econ., 66:269–285

Harker PT (1989) The art and science of decision making: The analytic hierarchy process: Applications and studies. Springer-Verlag

Harri IK (2002) Accounting in customer–supplier relationships: Developing cost management in customer–supplier relationships: Three case studies. In: Proceedings of the 3rd International Conference on New Directions in Management Accounting, Brussels, Belgium, 2:699–716

Holmlund-Rytkönen M, Strandvik T (2005) Stress in business relationships. J. Bus. Ind. Mark. 20(1):12–22

Hsieh FS (2008) Holarchy formation and optimization in holonic manufacturing systems with contract net. Automatica 44(4):959–970

IEEE P1516 Draft standard for modeling and simulation (M&S) high level architecture (HLA)—framework and rules

IEEE P1516.1 Draft standard for modeling and simulation (M&S) high-level architecture (HLA)—federate interface specification

IEEE P1516.2 Draft standard for modeling and simulation (M&S) high-level architecture (HLA)—object model template (OMT) specification

Koestler A (1967) The ghost in the machine. Arkana, London

Lauras M, Parrod N, Telle O (2003) Proposition de référentiel pour la notion d'entente industrielle: trois approches dans le domaine de la gestion des chaînes logistiques. RFGI, 22(4):5–30

Mekaouche L (2007) Pilotage holonique auto-organisé de réseaux logistiques : validation par modélisation et simulation distribuée. PhD, Université Paul Cézanne – Marseille, France

Mesarovic MD, Macko D, Takahara J (1970) Theory of hierarchical multilevel systems. Academic Press, New York

Nesheim T (2001) Externalization of the core: Antecedents of collaborative relationships with suppliers. Eur. J. Purch. Supply Manag., 7(4):217–225

Nishi T, Konishi M, Hasebe S (2005) An autonomous decentralized supply chain planning multistage production processes. J. Intell. Manuf., 16(3):259–275

Ounnar F (1999) Prise en compte des aspects décision dans la modélisation par réseaux de petri des systèmes flexibles de production. PhD, Institut National Polytechnique de Grenoble – Laboratoire d'Automatique de Grenoble, Grenoble – France

Ounnar F, Pujo P (2001) Décentralisation des mécanismes de pilotage de la relation donneurs d'ordres/fournisseurs. In actes du 4e Congrès International de Génie Industriel, France, 2: 1175–1185

Ounnar F, Pujo P, Mekaouche L et al. (2007) Customer–supplier relation-ship management in an intelligent supply chain network. Prod. Plan. Control, 18(5):377–387

Ounnar F, Pujo P, Mekaouche L et al. (2009) Integration of a flat holonic form in an HLA environment. . J. of Intell. Manuf., 20(1):91–111

Pujo P, Ounnar F (2007) Vers une approche holonique des systèmes mécatroniques complexes: proposition d'un système de pilotage auto organisé et isoarchique. JESA, 41(6):673–706

Pujo P, Broissin N, Ounnar F (2009) PROSIS: An isoarchic structure for HMS control. Eng. Appl. Artif. Intell. (Int. J.), doi:10.1016/ j.engappai.2009.01.011

Saaty TL (1980) The analytic hierarchy process. McGraw-Hill, New York

Smith R (1980) The contract net protocol. High-level communication and control in a distributed problem solver. IEEE Trans. Comput., 29(12):1104–1113

Smart A, Harrison A (2003) Online reverse auctions and their role in buyer–supplier relationships. J. Purch. Supply Manag., 9:257–268

Telle O, Thierry C, Bel G (2004) Aide à la coopération au sein d'une relation Donneur d'Ordres/Fournisseur dans le secteur aéronautique: Un outil de simulation. JESA, 38(1–2):7–36

Timmerman E (1986) An approach to vendor performance evaluation. JPMM, winter

Toolea T, Donaldsonb B (2002) Relationship performance dimensions of buyer–supplier exchanges. Eur. J. Purch. Supply Manag., 8:197–207

Trentesaux D (2007) Les systèmes de pilotage hétérarchiques. JESA, 41(9-10):1165–1202

Van Brussel H, Wyns J, Valckenears P et al. (1998) Reference architecture for holonic manufacturing systems: PROSA. Comput. Ind., 37(3):255–276

Vargas LG (1990) An overview of the analytic hierarchy process and its applications. Eur. J. Oper. Res., 48(1):2–8

Vincke P (1989) L'aide multicritère à la décision. Université de Bruxelles, Belgique

Wedley WC (1990) Combining qualitative and quantitative factors–an analytic hierarchy approach. Socio Econ. Plan. Sci., 24(1):57–64

Chapter 7
Supply Chain Management Under Uncertainties: Lot-sizing and Scheduling Rules

A. Dolgui, F. Grimaud and K. Shchamialiova

Abstract This chapter addresses the problem of optimal lot-sizing and sequencing of manufacturing items under uncertainties. A state of the art is presented and a case study is developed. For the case study, the following factors are taken into account: processing and set-up times, random machine breakdowns and rejects. The goal is to minimize the cost and maximize the probability of a desired output for a given period. For the considered case, first, a mathematical model of the problem and some heuristic and metaheuristics approaches are discussed. The first approach decomposes the initial task in two sub-problems: travelling salesman and knapsack problems. An iterative optimization procedure is proposed, based on this decomposition. Then, a genetic algorithm is suggested and detailed. Several test and simulation results are reported.

Keywords Supply chain, lot-sizing, scheduling, sequencing, uncertainty

7.1 Introduction

In the face of strong market competition, each company tries to find a strategy to produce goods of high quality cheaply, on time and in sufficient quantity to satisfy customer demands. To reach these objectives several production and operations management problems should be solved efficiently. One of them incurred often is lot-sizing and sequencing of manufacturing items in a supply chain.

There are many definitions of a *supply chain* in the literature (for example, (Lee and Billington, 1993; Pochet and Wolsey, 2006). In our opinion a supply chain is the network of enterprises destined to transform raw materials into finished products ready for delivery to the end customers. The supply chain consists of suppliers, companies, buyers, vendors, customers and others who associated with them (see Figure 7.1).

A. Dolgui (✉), F. Grimaud and K. Shchamialiova
Centre for Industrial Engineering and Computer Science, Ecole des Mines de Saint Etienne
158 cours Fauriel, 42023 Saint Etienne cedex 2, France
e-mail: dolgui@emse.fr

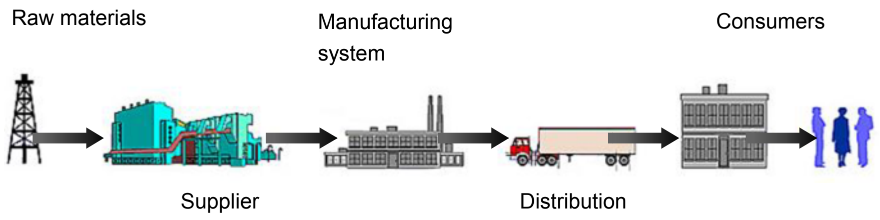


Fig. 7.1 Supply chain

Supply chain management (SCM) is the process of planning, coordination and executing the operations of supply chain with the goal to satisfy customer demands as efficient as possible. For more information, see Giard (2003) and Axsater (2006).

There are three levels of SCM decisions: strategic, tactical and operational. The decisions taken at the strategic level set the conditions for decision-making at the tactical and operational levels. The strategic level is a level of long-term decisions such as:

- geographic location, quantity and size of production facilities (suppliers, distribution centres, etc.);
- set of items to produce;
- suppliers to use;
- kinds of transport for the raw materials, and final products;
- which information technologies to use in support of supply chain operations.

The tactical level includes medium-term decisions: quantity and location of inventories, how and where to manufacture the products to minimize their final cost, etc.

The operational level of SCM consists of short-term day-to-day detailed production planning, quality control, production scheduling for each manufacturing facility and demand planning. The decisions at this level are made taking into account the strategic and tactical considerations. Note that the distance between tactical and operation levels is often very small and depends on the type of items produced. Frequently, the tactical and operational levels are considered together, because the goal of SCM decisions made at these levels is to achieve the most efficient work at the lowest cost.

In this chapter, we consider the processes occurring at the lower SCM levels (tactical and operational), namely the lot-sizing and sequencing problem occurring in the flow-shop manufacturing systems.

The ideal production strategy consists in processing part by part (the lot is equal to one part): in this case when the demand level for the product is reached we can stop the production. Unfortunately, this strategy (and in more general cases any lot-for-lot policy) can be applied only for simple production systems. Indeed, if customer demand should be satisfied immediately upon arrival (otherwise it will be lost), then the enterprise has to make inventories. The level of this inventory should be neither too small nor too large. In the former it may be insufficient to cover all demands. In the latter, the holding cost may be very high. Generally there are the set-up times or (and) costs before each production run or when changing the type of product. A set-up time is necessary for preparation of the line. If the cost (or time) of set-up is excessive, it may have a negative effect on the total cost (or time). That's why, for minimizing the total production time or (and) cost, the items of the same product should be processed in batches. So the production planning includes the following decisions: choosing a set of products to manufacture, sizes of batches (lots) for each product, defining periods when the products will be processed or the sequence of batches in case of single period and the machines where these processes will occur. In the literature, these problems are known as *lot-sizing* and *scheduling*. In the following sections, we present them, their classification and some particular cases.

The lot-sizing and scheduling techniques should give answers to the following questions:

- Which product(s) to produce?
- How many units of each product should be processed?
- What machine(s) should handle each product and in what order?
- When should the production begin?

The production systems can be classified by type (flow-shop, job-shop, etc.), number of stages (levels), variety of items produced, etc. The complexity of the lot-sizing and schedulingproblems depends directly on this. In addition, there are many random factors (uncertainties), which should be taken into account, for example:

- When the fabrication is launched, the breakdowns of some machines or elements of the manufacturing system may appear. Sometimes, faults can be neglected (small duration or repaired immediately). But, often it is necessary to consider possible breakdowns during the production plan design.
- Generally there are no perfect machines, so in lot-sizing and scheduling models the possibility of rejects (random yield) should be also taken into account.
- Often, when developing the production planning we don't know the exact level of customer demand for future periods. In this situation, it is necessary to make an estimation (demand forecast).
- The time of delivery also may be random.

In this chapter, we discuss the problems of lot-sizing and sequencing under uncertainties for a single machine or a flow-shop. There is a vast number of papers that deal with problems of this kind; among others we can cite the surveys of Yano and Lee (1995), Drexl and Kimms (1997), Singh *et al.* (1998) and Grosfeld-Nir and Gerchak (2004).

The remainder of this chapter is organized as follows. Sect. 7.2 gives a survey of the more popular lot-sizing and scheduling methods. First, some standard models of lot-sizing and scheduling with their extensions are reported. Then, the different kinds of uncertainties and their influence on lot-sizing and scheduling techniques are presented. At the end of this section, the methods most often applied to resolve the foregoing lot-sizing and scheduling problems under uncertainties are discussed. In Sect. 7.3, we present a case study. It concerns a particular lot-sizing and sequencing problem under uncertainties (random machine breakdowns and item rejects). We suggest a mathematical model and two resolution methods: dynamic programming and genetic algorithm (GA). At the end of this section there are some experimental results and simulations with ARENA. Finally, Sect. 7.4 presents some conclusions and perspectives for future research.

7.2 Short Presentation of Lot-sizing and Scheduling Problems

Lot-sizing methods are developed to decide whether and how many items (the lot size) to produce of a particular product for a given horizon. *Lot* is a batch of the items of the same type. The complexity of a lot-sizing problem depends on the considered production system. The classification of the lot-sizing problems is mainly based on the characteristics of the production systems. Below we describe the most important of them.

The *planning horizon* is the amount of time when all manufacturing operations will be carried out. The planning horizon may be *finite* or *infinite*. Often the finite planning horizon is divided into periods. There are two categories of lot-sizing problems in terms of the length of periods: *small-bucket* and *big-bucket* problems. For the small-bucket case, only one unit can be processed during a single period. In the case of big-bucket problems, the time periods are rather long, so it is possible to produce multiple items. The lot-sizing problems can be also considered for a one-period case, i.e., it is necessary to search for the lot size for only one period.

The production system may be *single* or *multi-level*. In the single-level case, the final product is simple. Its bill of material is of a single level. Therefore, to obtain the final product only one production stage is necessary. Multi-level systems are much more complex and more difficult to manage.

The number of final products in the production system is also an important characteristic. From this point of view, there are also two kinds of production systems: *single-* and *multi-item*. A single–item system fabricates only one type of final product, while in the multi-item case there are a certain number of final products.

When there are no limitations on the resources, the problem is *uncapacitated*. The problem is *capacitated* if the capacity and constraints are explicitly stated.

Demand may be *static* or *dynamic*. Static demand means that there are not any changes of demand values during the planning horizon, while the value of dynamic demand may change over time. If before beginning production, the value of demand (static or dynamic) is known, we deal with *deterministic* demand; otherwise, if we don't know the exact value of demand and certain probabilities are used, then we deal with *stochastic* demand.

Usually, when we pass from a product to another, an additional cost or (and) time of set-up is generated. Generally, this is because the necessity to prepare the machines for manufacturing the components of a new type, equipment adaptation and supply of required materials. If the set-up cost (or set-up time) for the current period is independent of the sequence of products and decisions made in previous periods, the corresponding problem is named *sequence-independent*. Otherwise, this is a *sequence-dependent* problem. Almost all of planning problems include the set-up times (costs).

The goal of scheduling problems is to find the order to pass the products through the machines to optimize an objective function (for example to minimize total time of production). We can classify the scheduling problems starting from the production system configuration. There are five fundamental types of the system configurations: single machine, parallel machines, flow-shop, job-shop and open-shop.

In the case of a single machine everything is simple: all units should be processed by this machine. The problem is to define an order (sequence) of items (lots) and in the multi-period case – processing period(s) for each unit. The machine can process only one unit at the same time.

Parallel machines execute the same operations, so each product can be processed on any of the machines. But despite this, the processing time can be different for different machines. Three types of parallel machines can be differentiated:

- *Identical machines*: the processing time of a given item is the same for all machines.
- *Uniform machines*: each machine j has a "speed" s_j, length of the processing period of the unit (job) i on the machine j is equal to the processing time t_i divided by speed s_j. In this case we can say that processing times are proportional.
- *Unrelated machines*: processing times t_{ij} of each unit (job) i on each machine j are given, moreover they are independent from each other.

An example of a production system with parallel machines is shown in Figure 7.2.

Flow-shop is a production line where n products should be processed by a set of k machines, and all these machines should be visited in the same order by all products. The processing times t_{ij} (for the product $i, \quad i = 1,...,n$ on the machine $j, \quad j = 1,...,k$) are given. If the condition $t_{ij} > 0$ is satisfied for each $i, \quad i = 1,...,n$ and $j, \quad j = 1,...,k,$ then the considered line is *pure* flow-shop. If there are processing times equal to zero, this flow-shop is *generalized*: if, for example, $t_{is} = 0$ then the product i will skip the operations on the machine s. If the sequence of units remains unchanged for all machines in the line, this is called flow-shop of *permutations*. Usually, the problem of minimizing the production time on a flow-shop line is NP-hard if the number of machines is more than two (nevertheless, note this is not always the case). There are several specific cases of three-machine flow-shops when the problem is not NP-hard.

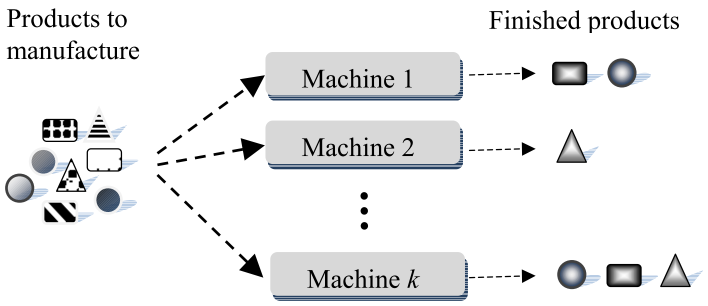


Fig. 7.2 An example of the system with k parallel machines

In a *job-shop* production line, there are n different products to manufacture and k machines. Each product (job) has its own sequence of operations, so the jobs shouldn't pass through all machines. For each job the sequence of operations is fixed. In Figure 7.3 an example of a job-shop system is presented. As you can see, the sequence of operations for the first job is $(1,3,4,5)$, this sequence is $(1,4,6)$ for the second and $(3,5,6,4,2)$ for number three. The job-shop problems are also NP-hard in most cases.

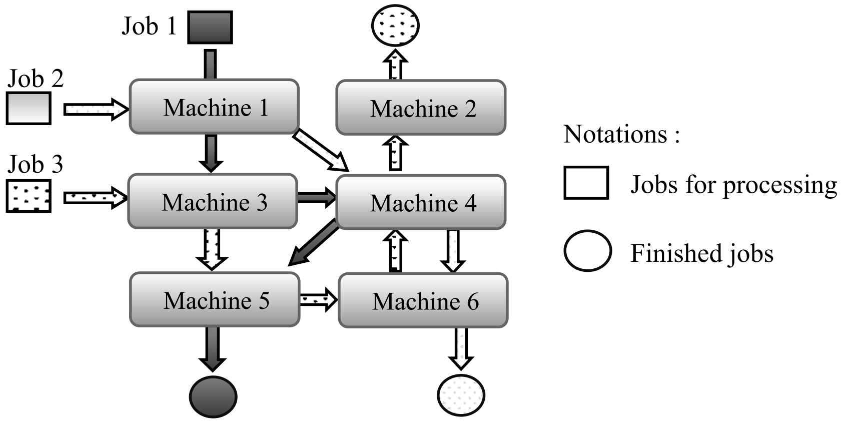


Fig. 7.3 An example of a job-shop with six machines and three jobs

Open-shop is the specific case of job-shop. The difference is that there are no ordering constraints on operations. Each product (job) should be treated by a certain set of machines, but the order of operations is absolutely not constrained.

For a survey of different scheduling problems with set-up times and/or costs (see Allahverdi *et al.*, 1999, 2008). In these surveys, the reader can find the literature concerning all three listed types of production systems. Moreover, Kis and Pesch (2005) proposed a review of exact methods for the hybrid flow-shop (here on each stage there are several parallel machines, and every job can be processed by any of them).

In the next section, some of the most basic lot-sizing problems are described.

7.2.1 Basic Models and Extensions

One of the oldest and most fundamental inventory models is the *economic order quantity* (EOQ) introduced by Harris in 1913. The goal of this problem is to find *economic order quantity* Q — the level of inventory that minimizes the total inventory holding and ordering costs. There are the following assumptions:

- units of a single product will be processed;
- annual demand D is known and constant;
- planning horizon is infinite;
- all items are of the perfect quality;
- ordering cost c is known and constant; and
- holding h and purchase p per unit costs is also known and constant.

To find the optimal order quantity it is necessary to minimize the average annual total cost:

$$C_{total}(Q) = p \cdot D + c \cdot D/Q + h \cdot Q/2, \tag{7.1}$$

where $p \cdot D$ is the purchasing cost, $C_O(Q) = c \cdot D/Q$ is the ordering cost, and $H(Q) = h \cdot Q/2$ is the holding cost. A graphical representation of these costs is provided in Figure 7.4. The optimal solution of this problem is easy to find: it is sufficient to set the derivative of total cost function to zero and solve the obtained equation. After performing that, we obtain:

$$Q^{opt} = \sqrt{\frac{2c \cdot D}{h}}. \tag{7.2}$$

The *economic production quantity* (EPQ) is one of extensions of EOQ model. It was suggested by Taft in 1918. The difference between the EPQ and EOQ models is that for the EPQ policy the orders are received progressively during the manufacturing process. Here we also look for the balance between inventory holding and average ordering costs.

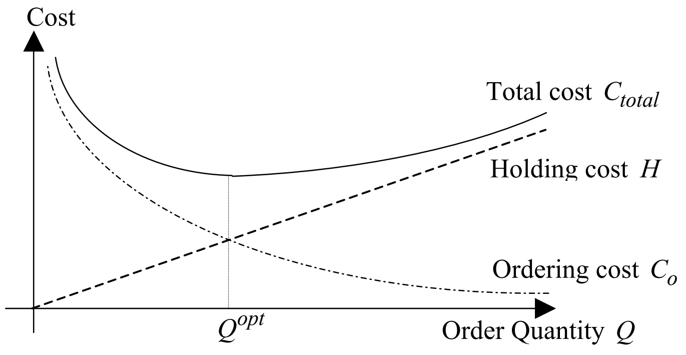


Fig. 7.4 EOQ model: the cost curves

Let the production rate be denoted by R. Then, using notation of EOQ, we obtain the following equation for the total cost:

$$C_{total}(Q^{EPQ}) = p \cdot D + c \cdot D / Q^{EPQ} + h \cdot Q^{EPQ} / 2(1 - D/R). \tag{7.3}$$

Performing the same mathematical developments as for the EOQ case, we obtain:

$$Q^{EPQ} = \sqrt{\frac{2c \cdot D}{h}} \cdot \sqrt{\frac{R}{R-D}}. \tag{7.4}$$

Note that if $R \to \infty$, then we have the EOQ model.

There are many other papers with different methods to optimize order or production quantities. Grubbstrom and Erdem (1999) developed an algebraic method (without derivatives) to solve the EOQ problem with backlogging. Cardenas-Barron (2001) extended this approach for the EPQ model with shortages. We can find similar approaches in Ronald *et al.* (2004) and Chang *et al.* (2005). Minner (2007) developed yet another approach for the EOQ and EPQ with backorders. The proposed method is based on cost comparison and again doesn't use the differential calculus. Teng (2009) proposed the *arithmetic–geometric–mean–inequality* theorem to find the optimal order quantity. This method is based on the fact that the arithmetic mean is greater or equal to the geometric mean. The author considers three cases: EOQ model with and without backorders and EPQ model with backorders.

The EOQ and EPQ models have several assumptions to simplify the real life situations. Nevertheless these models are a good base to deal with more sophisticated problems. We can find a large number of useful extensions of EOQ and EPQ in literature. For example, Khouja and Goyal (2005) developed an inventory model on the basis of the standard EOQ with the assumptions that component unit cost decreases over the time and the cycle time may change. In the papers by Eroglu and Ozdemir (2007) and Maddah and Jaber (2008), an EOQ model with defective items and shortages has been studied. Liao *et al.* (2009) considered the EPQ problem with imperfect production, which implies both imperfect repairs and out of control states.

The *Economic Lot Scheduling Problem* (ELSP) was proposed by Rogers in 1958. Unlike EOQ there are the multiple items and limited capacity. It was shown by Hsu in 1983 that ELSP is NP-hard. Descriptions of this problem, some extensions and optimization methods can be found in Yao and Elmaghraby (2001), Yao and Huang (2005), Chatfield (2007), Haji *et al.* (2008) and Teunter *et al.* (2009), etc.

The above problems have a stationary demand and an infinite planning horizon. In contrast, the one-item Wagner–Whitin model that will be considered has time-varying demand and finite time horizon. The notation for this model is as follows (see also Richter *et al.*, 2006):

- the planning horizon is divided on T discrete periods;
- there is not any inventory at the beginning of processing;
- c is the ordering cost for procurement and processing, constant for all periods;
- h is the per unit per period inventory holding cost;
- d_t is the order for the resources at the period t, $t = 1,...,T$; and
- I_t is the inventory at the end of period t, $t = 1,...,T$.

The goal is to find the production quantities $\left(x_1, x_2, ..., x_T\right)$ for all periods, which minimize the total cost (inventory holding and ordering costs):

$$\sum_{t=1}^{T} (c \cdot \operatorname{sign} x_t + h \cdot I_t) \to \min, \tag{7.5}$$

$$\begin{aligned} &I_t = I_{t-1} + x_t - d_t, \quad t = 1,...,T, \\ &I_0 = 0, \quad x_t, I_t \geq 0, \end{aligned} \tag{7.6}$$

where $\operatorname{sign} x = \begin{cases} +1 & x \succ 0, \\ 0 & x = 0. \end{cases}$

Wagner and Whitin proposed an optimization procedure to solve this problem. This procedure is based on a dynamic programming approach.

The standard *capacitated lot-sizing problem* (CLSP) supposes a known dynamic demand, finite planning horizon, capacity restrictions and set-up times. The goal of the problem is to find the production plan to minimize the total cost. Here, we have the following notation:

- number of periods T;

- number of products n;
- size of lot i in period t is equal to x_{it}, $i=1,...,n$, $t=1,...,T$;
- inventory volume I_{it} of item i at the end of period t, $i=1,...,n$, $t=1,...,T$;
- capacity R_t available in period t, $t=1,...,T$;
- demand d_{it} for item i in period t, $i=1,...,n$, $t=1,...,T$;
- unit production cost c_{it} of product i in period t, $i=1,...,n$, $t=1,...,T$;
- set-up cost s_{it} of item i if processed in period t, $i=1,...,n$, $t=1,...,T$;
- unit holding cost h_{it} for item i and period t, $i=1,...,n$, $t=1,...,T$;
- resource consumption a_i for component i, $i=1,...,n$; and
- upper bound on production quantity $B_{it}=\sum_{p=t}^{T} d_{ip}$ for product i at period t, $i=1,...,n$, $t=1,...,T$.

Let $y_{it}=\begin{cases}1, & \text{if item(s) of type } i \text{ is(are) produced in period } t\\ 0, & \text{otherwise}\end{cases}$, where $i=1,...,n$ and $t=1,...,T$.

Then, the mathematical formulation of this problem is the following:

$$\sum_{i=1}^{n}\sum_{t=1}^{T}\left(s_{it}\cdot y_{it}+c_{it}\cdot x_{it}+h_{it}\cdot I_{it}\right)\to \min \tag{7.7}$$

subject to:

$$\sum_{i=1}^{n} a_i x_{it}\le R_t,\quad t=1,...,T, \tag{7.8}$$

$$\begin{aligned} & x_{it}+I_{i,t-1}-I_{it}=d_{it},\quad i=1,...,n,\quad t=1,...,T,\\ & x_{it}\le B_{it}\cdot y_{it},\quad i=1,...,n,\quad t=1,...,T,\\ & x_{it}\ge 0,\quad I_{it}\ge 0,\quad i=1,...,n,\quad t=1,...,T.\end{aligned} \tag{7.9}$$

The CLSP is NP-hard. A broad survey of this problem is reported in Karimi *et al.* (2003).

There are some papers that study the problems of lot-sizing and scheduling simultaneously. In Emmons and Mathur (1995), a no-wait flow-shop line is considered. The goal is to minimize makespan. "No-wait" means that there is not any delay between the ending of one operation and the beginning of the subsequent one. There are n jobs to produce, divided into m groups. All jobs within the group have the same processing time on each machine. There are also set-up times between different products. The authors formulated and proved some theorems to reduce the size of problem. Lee *et al.* (1997) compared genetic and heuristic algorithms intended to solve this problem, but only for the flexible flow-shop lines. Potts and Kovalyov (2000) reviewed models that integrate scheduling and lot-sizing decisions for most production line configurations. Zhu and Wilhelm (2006) provided another overview of literature that concerns combined lot-sizing and scheduling problems for different configurations of production lines. Tempelmeier and Buschkuhl (2008) proposed an approach to a multi-item, multi-machine lot-sizing and scheduling problem with set-up times.

Many kinds of uncertainties should be taken into account when real-life lot-sizing and scheduling problems are considered. In the next subsection, some of them will be presented.

7.2.2 Lot-sizing and Scheduling Under Uncertainties

Uncertainty in the production process is a well-known phenomenon common to many manufacturing systems. This can adversely influence value-adding processes. This may concern machine breakdowns, defective items (rejects), uncertain demands, and random events during transportation, etc. According to Tajbakhsh *et al.* (2007) there are three main types of uncertainties:

1. stochastic lead-time;
2. uncertainties in supply quantity;
3. uncertainty in purchasing price.

The first two uncertainties will be looked at in depth.

7.2.2.1 Stochastic Lead Times

Lead-time is a period of time between the initiation of any process of production and the completion of that process. Lead-time uncertainty means that the actual lead-time may differ from planned. This kind of uncertainty exists throughout the supply chain (see Figure 7.5).

To minimize the effect of uncertainties in lead-time, companies use stocks. The level of stock can have a significant impact on the business: too high a stock level can increase holding costs and consequently the final price of the product; low stock level leads to unsatisfied customer demand and lost sales. To control the stocks in the most industrial applications the *material requirement planning* (MRP) approach is used.

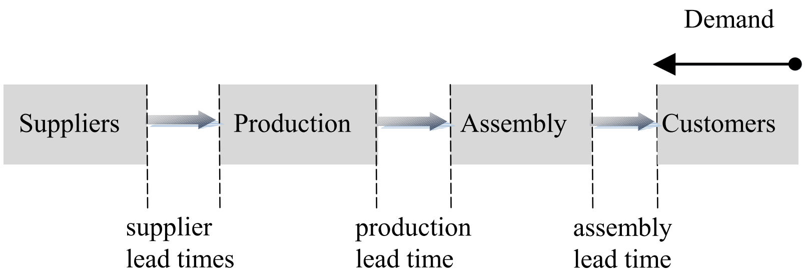


Fig. 7.5 Lead-time uncertainty in a supply chain

Classic MRP develops production plans and schedules with help of *bills of materials* (the list of raw materials and components, and their quantities needed for manufacturing one final product). Lead times are used to calculate the dates for raw material and component orders taking into account current and projected inventory levels. It is assumed that demand and lead times are known. In reality these deterministic assumptions of MRP are too restrictive because of demand and lead-time uncertainties. Therefore, safety stocks (extra units of inventory) or/and safety lead times were introduced in MRP to cope with uncertainties.

There are some papers which describe the problems of MRP under uncertainties and propose methods to calculate optimal safety stocks or lead times (Molinder, 1997, Wilhelm and Som, 1998, Fong *et al.*, 2000, Dolgui and Louly, 2002, Louly and Dolgui, 2004, 2009, Gurnani and Gerchak, 2007 and Dolgui *et al.*, 2008a). A state of the art for the supply planning problems under uncertainties in MRP environment is presented in Dolgui and Prodhon (2007).

Dolgui and Louly (2002) searched for the optimal values of the planned lead times (safety lead times) for the MRP method. The distributions of the random lead times are known. The goal is to find the planned lead times to minimize the total of expected backlogging and holding costs. Their model contains the following assumptions:

- Demand D per period is constant, and ordered quantity is known.
- Unit holding cost h per period is known.
- Unit backlogging cost b per period is known.
- A lead-time L is a random variable with known distribution $p_i = pr\left(L = i\right), \quad i = 1,...,u$, u is the maximum lead-time value.

For the one-period model the formulation of the cost is the following:

$$C(T,L) = hD(T-L)^{+} + bD(L-T)^{+} \tag{7.10}$$

and the expected cost:

$$EC(T) = hD\sum_{i=1}^{T}(T-i)p_i + bD\sum_{i=T}^{u}(T-i)p_i. \tag{7.11}$$

The optimal value of T can be found by minimizing Eq. 7.11.

An example of where lead-time uncertainty may appear is when there is a possibility of random machine breakdowns. After a breakdown the machine is unavailable for a certain period due to repairs. Generally, the time to repair is also random. Below, we present a renewal process model, which is intended for evaluate the work of this imperfect production system. This model was studied in Dolgui (2002). Random breakdowns may interrupt the normal processing. A scheme of a renewal process with breakdowns and repair intervals is shown in Figure 7.6.

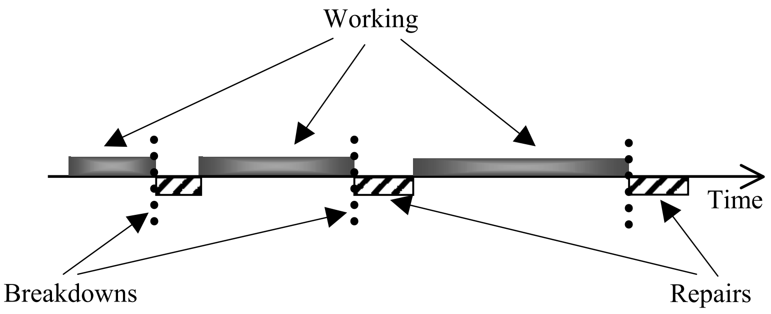


Fig. 7.6 An example of a renewal process

Cumulative working time is the total time the system works normally. It is an important concept for renewal process models. Note that the cumulative working time is also a random variable. However, we can find the probability distribution of this variable.

The total cumulative working time h_t for the time interval $[0, t]$ is the sum of all periods when the production system functioned well (working periods in Figure 7.6). The next characteristic is used to estimate the cumulative working time:

$$A(s,t) = p(h_t < s) \tag{7.12}$$

$P(h_t < s)$ is the probability that the cumulative working time will be inferior to a certain value s for the time interval $[0, t]$. In Dolgui (2002), there are detailed instructions on how this probability can be obtained. We present here only the final equation:

$$A(s,t) = \sum_{\nu=0}^{\infty}\left[G^{*\nu}(t-s) - G^{*\nu+1}(t-s)\right] \times F^{*\nu+1}(s) \tag{7.13}$$

where $G(\cdot)$ is a distribution function of the repair time, $F(\cdot)$ is the distribution function of the time between breakdowns. The index $*\nu$ means the ν-th convolution integral of the correspondent function and can be calculated as follows:

$$F^{*\nu}(t) = \int_0^t F^{*(\nu-1)}(t-y)dF(y), \quad F^{*0}(t) \equiv 1. \tag{7.14}$$

If λ is a breakdown rate and μ is a repair rate, then

$$F^{*\nu}(s) = 1 - \exp(-\lambda s) \times \left\{ 1 + \frac{\lambda s}{1!} + \frac{(\lambda s)^2}{2!} + \ldots + \frac{(\lambda s)^{\nu-1}}{(\nu-1)!} \right\}. \tag{7.15}$$

Replacing λ by μ and s by $t-s$ in this equation, we can obtain a similar expression for $G^{*\nu}$. Substituting the obtained formulas into Eq. 7.13, we obtain:

$$A(s,t) = \exp\left\{-\left(\lambda s + \mu(t-s)\right)\right\} \times \sum_{\nu=1}^{\infty} \left(\left((\lambda s)^{\nu} / \nu! \right) \sum_{j=0}^{\nu-1} \left(\mu(t-s)\right)^{j} / j! \right) \tag{7.16}$$

There are many other reasons to have random lead-time. For example, Cakanyildirim *et al.* (2000) and Febbraro *et al.* (2001) considered inventory systems with random lead-time, because of random (variable) processing time for each item.

7.2.2.2 Uncertainties in Supply Quantity

A random yield is another type of uncertainty. Presence of random yield can significantly complicate production planning. There are many domains where yield is random, for example, in electronic and chemical industries. If yield is random, the output of the system may be different from the input. There are several methods to model yield uncertainty:

1. Bernoulli process: here the probability p that a given item is of sufficient quality is known, and the quantity of good items d in a lot of size x follows a binomial distribution, i.e., the probability to obtain exactly d good items from x processed is:

$$P(d \mid x) = C_x^d \cdot p^d \cdot (1-p)^{(x-d)}, \text{ where } C_x^d = x!/d!(x-d)!. \tag{7.17}$$

Processing of one item is assumed to be independent from processing of others (see Singh *et al.*, 1988, Teunter and Flapper, 2003, and Dolgui *et al.*, 2005).
2. Stochastically proportional yield is the most generally used approximation. (This kind of yield modelling can be found in Yao, 1988, Gerchak *et al.*, 1994, Agnihothri *et al.*, 2000, Wang and Gerchak, 2000, Li *et al.*, 2008, Haji *et al.*, 2008, etc.) Here the fraction of the good components is known, moreover it is independent from the lot size: x is the lot size, α is a random variable, $\alpha \in [0,1]$. The resulting yield is equal to $x \cdot \alpha$.
3. Interrupted geometric: in this method there is one point in time (a moment when "in-control" process becomes "out-of-control") such that all components manufactured before are of good quality, and those processed after are rejects. Let $1-\theta$ be the probability that the process changes its state from "in-control" to "out-of-control". Then, the probability distribution of the number d of good units when x units were processed is calculated as follows:

$$P(d,x) = \begin{cases} (1-\theta)\cdot\theta^d, & d = 0,1,2,\ldots,x-1, \\ \theta^d, & d = x. \end{cases} \tag{7.18}$$

The probability $P(d,x)$ is independent of x when $d < x$. (The lot-sizing models with *interrupted geometric yield* distribution can be found in Grosfeld-Nir and Gerchak, 1996, Guu and Liou, 1999, Guu and Zhang, 2003, etc.)
4. The yield uncertainty is a result of random capacity. Therefore, the obtained quantity is the minimum of the input quantity and realized capacity, which is random.

Several authors propose models with a general yield distribution, i.e., any distribution can be used. Zhang and Guu (1997) examined the properties of cost function for a multiple lot-sizing problem with constant demand and general random yield.

In the literature, the lot-sizing problems with uncertainties were reviewed by many authors. Yano and Lee (1995) provided a survey on different production systems with random yield: single-stage continuous and discrete, single- and multi-product, assembly, with rework, etc. Gerchak *et al.* (1994) proposed a formulation of a lot-sizing problem for an assembly system with random yields. They consider two cases: when distributions of random yields are identical and not.

Often problems contain both types of these uncertainties mentioned. For example, in Radhoui *et al.* (2009), a machine is subject to random failures and, in addition, can produce defective items.

Random demand is another widely used type of uncertainty considered in many publications. Usually, the probability distribution function is known (Gerchak and Parlar, 1986, Grosfeld-Nir and Gerchak, 2004, and Li *et al.*, 2008).

7.2.3 Optimization Techniques

Existing methods of lot-sizing and scheduling problems can be divided into three groups: exact, ε-approximate methods and heuristics. Exact methods render the optimal solutions for a problem, but if the considered model is close to an actual industrial case, it is often NP-hard. So it is not efficient to apply an exact method because the computation time becomes unreasonable. The ε-approximate methods can find a solution close to optimal, and the maximum possible deviation ε of the approximate solution from the optimal is known. Unfortunately in the majority of cases the complexity of ε-approximate approaches is similar to exact methods, so the most frequently used methods are heuristics and metaheuristics.

7.2.3.1 Exact Methods

Exact methods can be efficient for comparatively simple problems, for example EOQ (see Cardenas-Barron, 2001, and Teng, 2009).

Inderfurth *et al.* (2006) proposed a polynomial dynamic programming algorithm to solve a deterministic one-product lot-sizing and scheduling problem with rework of defective items. There are two stages in the processing of a batch: work and rework; a set-up time before processing each batch and a set-up time for passing from the work to rework stage. The proportion of defective items is known. There is a limited time for holding the defective items, after that they are no longer considered reworkable. The goal is to find the sizes of lots and determine positions of components, which will be reworked to satisfy demand and minimize the set-up, inventory holding, rework and shortage costs. In Inderfurth (2009), this dynamic programming approach was extended for some other problems of this type.

7.2.3.2 ε-approximate Methods

Kubiak *et al.* (2002) presented fully polynomial approximate schemes (FPASs) for the problem of minimizing completion time variance of n jobs on a single machine.

Kacem and Mahjoub (2009) proposed a FPTAS (T = time) algorithm for minimizing the processing time for n jobs on a single machine, but the machine can be unavailable during certain periods. The authors also provide an interesting analysis of complexity showing that the complexity of this algorithm is $O\left(n^2 / \varepsilon^2\right)$.

Dolgui *et al.* (2008b) considered lot-sizing and sequencing problems on a single machine. There are n types of products for manufacturing. The production is liable to random breakdowns and rejects. The set-up times to switch from producing one type of item to another are taken into account. The problem is to find the sequence of lots and their sizes to minimize total time (or makespan) in the first case and the total cost of delays in the second. An algorithm FPTAS based on the dynamic programming approach was suggested for this particular case.

7.2.3.3 Genetic Algorithms

Chatfield (2007) proposed a genetic approach for the ELSP. In the paper by Sikora (1996), a GA for the multi-product lot-sizing and scheduling problem on a flow line is also suggested. The system supposes sequence-dependent set-up times, due dates and capacity constraints. The finite planning horizon is divided into a certain number of equal periods. The author shows that the parameters of the GA such as the population size exert essential influence on the performance of the algorithm. Hernandez and Suer (1999) also studied the impact of different characteristics of the GA on the results. They consider a single-item one-level lot-sizing problem. (Analogous studies for different problems are performed by Lee *et al.*, 1997, Yao and Huang, 2005 and Gaafar, 2006).

Nevertheless, as aforementioned, the heuristic and metaheuristics are generally fast, but when applied, estimating how close the obtained approximate solution is to the optimal is exceedingly difficult.

7.3 A Case Study

A case study for a joint lot-sizing and sequencing problem under uncertainties is considered here. This is based on a real-life problem in the electronics industry, which has been examined by the authors. This problem consists in defining the sizes of lots and the sequence of these lots processed on the same production line. Machine breakdowns and part rejects are frequent and must be taken into account. This line supplies the next assembly shop. The items produced by the line are used in assembly of different products. The just-in-time policy is used. Thus, for a given planning horizon, it is necessary to optimize the number of good items of all types for the output of the line. Otherwise there is a risk that the entire assembly shop will be stopped.

7.3.1 Description of the Case Study

In Dolgui *et al.* (2005), a production line where several machines are placed sequentially is considered. In other words this is a permutation flow-shop. The line is meant for the manufacture of intermediate items of different types in lots. Remember that one lot is a set of items of the same product that pass through the line sequentially without any other product items inserted in between.

The system works under the following conditions:

- No more than one component may be manufactured by each machine simultaneously.
- To pass from processing of one lot to another it is necessary to consider an additional time – set-up time. Note: there is a set-up time before the first lot also.
- Machines can break down; in this situation, the manufacturing process should stop to repair the line. The characteristics of breakdown and reparation times are random.
- Machines can produce defective items. We can detect this only after processing on the last machine, so there are no intermediate control quality. Therefore, defective items are rejected only at the end of the line.
- There are no delays for product passing from one machine to the next.

The goal is to maximize the probability to have all needed on hand at the end of the planning horizon (the total time of line functioning cannot exceed a given value). Often in the literature this kind of objective function is called the maximizing of the *service level*. Note here the service level for all item types is considered simultaneously. To accomplish this, the following have to be optimized:

1. Sequence of lots: taking into account sequence-dependent set-up times, the order of item processing has a great influence on the total production time.
2. Sizes of lots: because of rejects and breakdowns we cannot determine the exact number of components to be released in production to cover all demands. Increasing the number of units does not necessarily increase the total probability to have all necessary items at the end of the planning horizon (even more the service level for the last lots can decrease drastically, because of limited time, i.e., fixed planning horizon).

Let's have n types of items for manufacturing. The number of machines in the line is equal to k. The next additional parameters are known:

- demand levels d_i, $i = 1,...,n$ for each product;
- value of the planning horizon T_0;
- machine processing per unit times t_i, $i = 1,...,n$ (since we are dealing with equivalent machines, the processing times for a given unit are the same for all machines at the production line);
- probability β_i of a minimum service level for the product i, $i = 1,...,n$;
- line set-up time $s_{i,j} > 0$, $i = 1,...,n$, $j = 1,...,n$, $i \neq j$ to switch from producing components of type i to components of type j; and
- line set-up time $s_{0,i} > 0$, $i = 1,...,n$ if we begin the manufacturing from items of the product i.

In this case study it is assumed that the set-up times satisfy the *triangle inequality*, i.e., $s_{i,q} + s_{q,j} \geq s_{i,j}$, $i,\ q, j = 1,...,n$. This fact leads to the conclusion that all the items of same product should be processed in exactly one lot. Figure 7.7 illustrates the processing of one lot without regard for rejects and machine breakdowns.

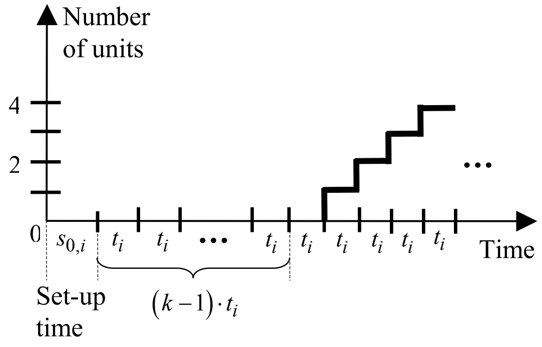


Fig. 7.7 The output for a lot

To explain the idea better, let lot i be the first lot at the line. Remember that there is a set-up time before producing each lot. After the set-up time, the first component of the lot enters the first machine. When the second machine processes the first component of the lot, the second component of the lot is treated by the first machine, and so on. So the manufacturing of the first item of lot i will be finished in $s_{0,i} + k \cdot t_i$ units of time. After that in t_i units of time the second item will leave the last machine, etc. Define the value $(k-1)\cdot t_i$ as the *loading time* for the product i, $i = 1,...,n$. A schematic drawing of the output is presented in Figure 7.8. Specific set-up and loading times precede the fabrication of each lot. Reparation occurs after each breakdown. It should be clarified that breakdowns may appear only during the processing of components. So, the processing of each lot includes: set-up time, loading and processing times, as well as time for repairing the line if a breakdown has happened.

Let the planned safety time T_{ps} be what we expect to spend on the recovery of the line after breakdowns during the planning period. Let T_{ps}^{act} be the actual repair time, and a random variable. Thus, if the planned safety time appears insufficient for all repairs (i.e., $T_{ps}^{act} > T_{ps}$), then several items of the last lot won't be processed (see Figure 7.8).

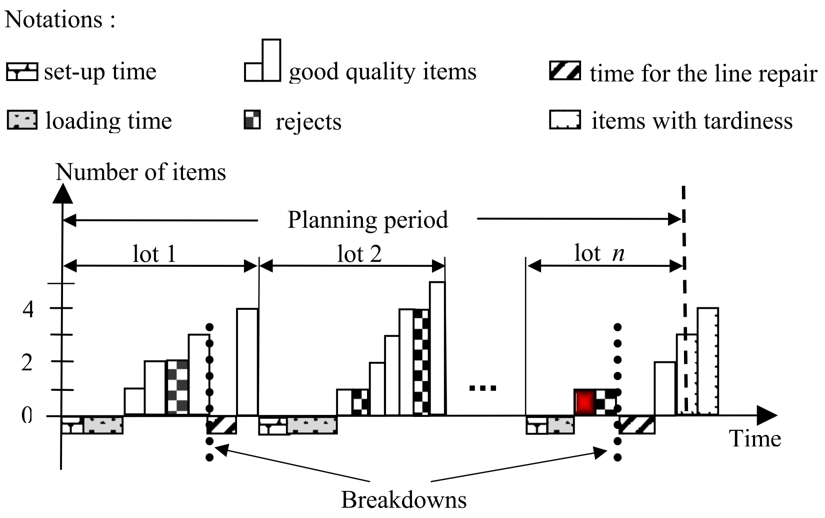


Fig. 7.8 An example of production sequence (output)

Let $x=(x_1,x_2,...,x_n)$ be the sizes of lots and $\pi=(i_1,i_2,...,i_n)$ be their sequence. Both are decision variables. Because inadequate products (rejects) can appear, the quantity x_i^g of good components obtained after processing x_i units of the lot number i can be less than x_i.

Therefore, in these terms, the optimization problem can be presented as follows:

$$P\left(x_i^g \geq d_i,\quad i=1,...,n \mid (\pi,x)\right) \to \max \tag{7.19}$$

subject to $\sum_{j=1}^{n} s_{i_{j-1},i_j} + (k-1)\sum_{i=1}^{n} t_i + \sum_{i=1}^{n} t_i x_i + T_{ps}^{act} \leq T_0.$

7.3.2 Representation of Uncertainties

As was mentioned earlier, we deal with two kinds of uncertainties: random yield and random lead-time. In this subsection, we will discuss both in more detail.

Suppose that the probabilities $p_{iq},\ i=1,...,n,\ q=1,...,k$ that a given unit of the product i processed on the machine q is of good quality are known. Since there are no buffers in between machines and no intermediate quality control, we can consider the line as a single machine. Therefore, the probability p_i that a finished product at the end of the line is of adequate quality is equal to:

$$p_i = \prod_{q=1}^{k} p_{iq}, \tag{7.20}$$

and $1 - p_i$ is the probability of obtaining a reject.

We deem that the quantity of quality x_i^g items in the lot of size x_i follows a binomial distribution, i.e.:

$$p_i(x_i^g \mid x_i) = C_{x_i}^{x_i^g} \left(1 - p_i\right)^{\left(x_i - x_i^g\right)} p_i^{x_i^g}. \tag{7.21}$$

In our problem, we should satisfy the demand d_i for each product.

Let $p^{>}(d_i \mid x_i)$ be the probability to obtain not less than d_i components of an acceptable quality of item i, taking into account the fact that x_i items were manufactured. Then

$$p_i^{>}(d_i \mid x_i) = \sum_{s=d_i}^{x_i} C_{x_i}^{s} \left(1 - p_i\right)^{\left(x_i - s\right)} p_i^{s}. \tag{7.22}$$

To describe breakdowns, the renewal process model (see Sect. 7.2.2.1) will be used. In this model, random breakdowns may interrupt the normal processing. As aforementioned, the time for line reparation is also a random value.

For our problem the following parameters are known:

- $1/u_i$ is the mean time to failure (MTTF) for the machine $i, \quad i = 1,...,k.$
- $1/r_i$ is the mean time to repair (MTTR) for the machine $i, \quad i = 1,...,k.$

Since there are no buffers in the line, failure U and repair R rates of the line (can be considered as one machine) will be calculated as follows:

$$U = \sum_{i=1}^{k} u_i, \tag{7.23}$$

$$R = \left(\sum_{i=1}^{k} u_i / r_i U \right)^{-1}. \tag{7.24}$$

Note that for this case study, the cumulative working time is known for each given vector $x = (x_1, x_2, ..., x_n)$. We will search for the probability that the actual repair time T_{ps}^{act} does not exceed the value of the planned safety time $p(T_{ps}^{act} \leq T_{ps})$.

After some transformations, we obtain:

$$A(s,t) = p(h_t < s) = p(t - h_t > t - s) = 1 - p(t - h_t \leq t - s). \tag{7.25}$$

Considering that $t - h_t = T_{ps}^{act}$ and $t - s = T_{ps}$, we derive the following expression for the actual time for line repairing:

$$\begin{aligned} p(T_{ps}^{act} \leq T_{ps}) &= 1 - A(s,t) \\ &= 1 - \exp\left\{-\left(Us + R\left(t-s\right)\right)\right\} \times \sum_{\nu=1}^{\infty} \left(\left((Us)^{\nu} \big/ \nu!\right) \sum_{j=0}^{\nu-1} \left(R\left(t-s\right)\right)^{j} \big/ j! \right). \end{aligned} \tag{7.26}$$

7.3.3 Objective Function

As in Dolgui *et al.* (2005), we will use here the following additional notation:

- $T^{+}(\pi, x) = \sum_{i=1}^{n-1} t_i x_i$ is the total processing time for the first $n-1$ lots.
- $T_{nr}(\pi, x) = T_0 - \left(\sum_{j=1}^{n} s_{i_{j-1}, i_j} + (k-1) \sum_{i=1}^{n} t_i + T^{+}(\pi, x) \right)$ is the time remaining for manufacturing items of the last lot and for all necessary line repairs.

To simplify the writing, let $T^{+} = T^{+}(\pi, x)$ and $T_{nr} = T_{nr}(\pi, x)$.
Then, the probability $P(\pi, x)$ of obtaining all required demands can be presented as follows:

$$P(\pi, x) = \prod_{i=1}^{n-1} p_i^{>}\left(d_i \mid x_i\right) \times p_n^{>}\left(d_n \mid x_n, T^{+}, T_{nr}\right), \tag{7.27}$$

where $p_n^{>}\left(d_n \mid x_n, T^{+}, T_{nr}\right)$ is the probability to obtain at least d_n items of good quality for lot n, which can be calculated subsequently:

$$p_n^{>}\left(d_n \mid x_n, T^+, T_{nr}\right) = \sum_{z=d_n}^{x_n} p_n^{>}(d_n \mid z) \times p_n^{+}\left(z \mid x_n, T^+, T_{nr}\right), \tag{7.28}$$

where $p^+\left(z \mid x_n, T^+, T_{nr}\right)$ is the probability that exactly z units from x_n planned (released or launched on the line) will be processed.

The probability $p^+\left(z \mid x_n, T^+, T_{nr}\right)$ can be determined as follows:

$$p_n^{+}\left(z \mid x_n, T^+, T_{nr}\right) = p_n^{>}\left(z \mid x_n, T^+, T_{nr}\right) - p_n^{>}\left(z+1 \mid x_n, T^+, T_{nr}\right), \tag{7.79}$$

where $p_n^{>}\left(z \mid x_n, T^+, T_{nr}\right)$ is the probability that we can process at least z units of the last product from x_n planned (released).

Note that for $z \le x_n$, the probability $p_n^{>}\left(z \mid x_n, T^+, T_{nr}\right)$ coincides with the probability $p\left(T_{ps}^{act} \le T_{nr} - t_n z \mid T^+ + t_n z\right)$ which was defined in the preceding subsection, where $T_{nr} - t_n z = T_{ps}$, and $T^t = T^+ + t_n z$ is the *objective processing time* of the line (reminder: a breakdown can only appear during a processing stage). To use the equations of the previous subsection, it is sufficient to put $T^t = s$ and $T_{ps} = t - s$. Therefore:

$$p_n^{>}\left(z \mid x_n, T^+, T_{nr}\right) = p(T_{ps}^{act} \le T_{ps} \mid T^t) = 1 - \exp\left\{-\left(T^t U + T_{ps} R\right)\right\} \times \\ \sum_{\nu=1}^{\infty}\left(\left(\left(T^t U\right)^{\nu} / \nu!\right) \sum_{j=0}^{\nu-1} \left(T_{ps} R\right)^j / j!\right). \tag{7.30}$$

7.3.4 Decomposition for Optimization

As shown in Dolgui *et al.* (2005), the following decomposition scheme can be applied. First, calculate the interval $\left[\underline{x}_i, x_i^+\right]$ of possible values for each size of lot $x_i,\ i = 1, \ldots, n$. As the probability of a minimum service level for each product is known, we can calculate the values:

$$\underline{x}_i = \min\left\{s \mid p_i^{>}(d_i \mid s) \geq \beta_i, \quad s = d_i, d_i + 1, \ldots\right\} \tag{7.31}$$

and

$$x_i^+ = \min\left\{\overline{x}_i, \underline{x}_i + \left(T_0 - \sum_{j=1}^{n} s_{i_{j-1}, i_j} - (k-1)\sum_{j=1}^{n} t_j - \sum_{j=1}^{n} t_j \underline{x}_j\right) / t_i\right\}, \tag{7.32}$$

where $\overline{x}_i = \min\left\{s \mid p_i^{>}(d_i \mid s) \geq 1 - \varepsilon, \quad s = d_i, d_i + 1, \ldots\right\}$ and ε is a sufficiently small value.

Hence, $\underline{x}_i$ is the minimum size of lot i, $i = 1, \ldots, n$ for which the probability of satisfying demand for this lot will be not less than β_i. Similarly, x_i^+ is the minimal possible size of lot i, $i = 1, \ldots, n$ for which the probability to satisfy the demand is close to 1.

Let Π be the set of all possible permutations π on the ensemble $\{1, 2, \ldots, n\}$. The Cartesian product $X = X_1 X_2 \ldots X_n$, where $X_i = \left\{\underline{x}_i, \underline{x}_i + 1, \ldots, x_i^+\right\}$, $i = 1, \ldots, n$ gives all possible combinations of lot sizes for the problem considered. Then, we have the following optimization problem:

$$P(\pi, x) = \prod_{i=1}^{n-1} p_i^{>}(d_i \mid x_i) \times p_n^{>}\left(d_n \mid x_n, T^+(\pi, x), T_{nr}(\pi, x)\right) \to \max, \tag{7.33}$$

where $(\pi, x) \in \Pi \times X$.

Let (π^*, x^*) be the optimal solution of this problem. We can detect that the sequence π has influence really only on the value $T_{nr}(\pi, x)$ (the time intended to manufacture the last lot including any troubleshooting on the line). Note that increasing the value $T_{nr}(\pi, x)$ leads to increasing the probability to finish the manufacturing of the last lot and all repairs by the end of the horizon, thus augmenting the service level. To find the optimal sequence π^* it is sufficient to resolve the following problem:

$$S(\pi) = \sum_{j=1}^{n} s_{i_{j-1}, i_j} \to \min, \tag{7.34}$$

where $\pi \in \Pi$. This is a modification of the well-known *traveling salesman* problem (see a proof in Dolgui *et al.*, 2008b). So each optimal solution of the asymmetric salesman problem gives the optimal solution π^*. There are many different methods to resolve this problem. Thus, we decided not focus our attention on this and go to resolving the second part of the problem considered (lot sizes).

Suppose that the optimal sequence of lots π^* was found using one of the known techniques and the lots are numbered according to the sequence found. Consequently, we can reformulate the problem as follows:

$$P'(x) = \prod_{i=1}^{n-1} p_i^{>}\left(d_i \mid x_i\right) \times p_n^{>}\left(d_n \mid x_n, T^+(x), T'_{nr}(x)\right) \rightarrow \max, \qquad (7.35)$$

where $x_i \in X_i, \quad i = 1,\dots,n-1$ and $T'_{nr}(x) = T_0 - \left(S\left(\pi^*\right) + (k-1)\sum_{i=1}^{n} t_i + T^+\left(x\right) \right)$.

Two methods to resolve this problem are developed: dynamic iterative programming and GA.

7.3.4.1 Iterative Optimization Procedure

Let

$$T_1 = T_0 - S\left(\pi^*\right) - (k-1)\sum_{i=1}^{n} t_i, \qquad (7.36)$$

then $T'_{nr}(x) = T_1 - T^+\left(x\right)$.

Definitions:

1. $\underline{w}_i = t_i \underline{x}_i$ and $\overline{w}_i = t_i x_i^+$ are the minimal and maximal times necessary for processing the lot $i, \quad i = 1,\dots,n.$
2. $\underline{V}_i = \sum_{j=1}^{i} \underline{w}_j$ is the minimal time necessary for manufacturing the first i lots, $i = 1,\dots,n-1$.
3. $\overline{V}_i = \min\left\{ T_1 - \sum_{j=i+1}^{n} \underline{w}_j, \ \sum_{j=1}^{i} \overline{w}_j \right\}$ is the maximal possible time that can be used for processing the first i lots, $i = 1,\dots,n-1$.

Using these notation we can present the problem in the following way:

$$P''(w) = \prod_{i=1}^{n-1} p_i^{>}\left(d_i \mid w_i / t_i\right) \times p_n^{>}\left(d_n \mid x_n, \sum_{i=1}^{n-1} w_i, T_1 - \sum_{i=1}^{n-1} w_i\right) \to \max, \tag{7.37}$$

where $w_i \in \{\underline{w}_i, \underline{w}_i + t_i, ..., \overline{w}_i\}$, $w_i \quad i = 1,...,n-1$.

In this case, we consider the time intervals w_i for manufacturing each lot i, $i = 1,...,n-1$. And if $w^* = \left(w_1^*, w_2^*, ..., w_{n-1}^*\right)$ is the optimal solution of this problem, then $x^* = \left(w_1^* / t_1, w_2^* / t_2, ..., w_{n-1}^* / t_{n-1}\right)$ is the optimal solution for the initial problem. Obviously, this is one modification of the *knapsack problem*. Let $V^i = \left\{V_1^i, V_2^i, ..., V_l^i\right\}$ be a set of values, each of which is the time that can be spent for manufacturing the first i lots. This set includes all possibilities.

The dynamic programming procedure is as follows (Dolgui *et al.*, 2005):

1. Calculate $H_1\left(V^1\right) = p_1^{>}\left(d_1 \mid V^1 / t_1\right)$ for each V^1, such that:

$$V^1 \in \left[\underline{V}_1, \overline{V}_1\right] = \left\{\underline{x}_1 t_1, \left(\underline{x}_1 + 1\right) t_1, ..., \min(x_1^+, x_1^{\max})\, t_1\right\},$$

where $x_1^{\max} = \left\lfloor (T^1 - \sum_{j=2}^{n} \underline{w}_j) / t_1 \right\rfloor$;

and set $i := 2$.

2. The set of values V^i is computed by the following:
   – for each V_j^{i-1}, $j = 1,...,\left|V^{i-1}\right|$ calculate the values $V_j^{i-1} + \underline{x}_i t_i$, $V_j^{i-1} + \left(\underline{x}_i + 1\right) t_i, ..., V_j^{i-1} + x_i^+ t_i$ and
   – values belonging to the interval $\left[\underline{V}_i, \overline{V}_i\right]$ are assigned to the set V^i.

   Calculate H_i for each V_j^i, $j = 1,...,\left|V^i\right|$ using the following formula:

$$H_i\left(V_j^i\right) = \max\left\{H_{i-1}\left(V_k^{i-1}\right) \times p_i^{>}\left(d_i \mid \left(V_j^i - V_k^{i-1}\right) / t_i\right) \;\middle|\; V_k^{i-1} \in V^{i-1}, V_j^i - V_k^{i-1} \in \left[\underline{w}_i, \overline{w}_i\right]\right\} \tag{7.38}$$

where $p^{>}\left(d \mid a\right) = p^{>}\left(d \mid \lfloor a \rfloor\right)$ for $a \in \Re_+$.

   Set $i := i + 1$. If $i \le n-1$ repeat step 2, else go to step 3.

3. Calculate the probability

$$P''(w^*) = \max\left\{H_{n-1}(V_j^{n-1}) \times p_n^{>}\left(d_n \mid x_n^+, V_j^{n-1}, T_1 - V_j^{n-1}\right) \;\middle|\; V_j^{n-1} \in V^{n-1}\right\}. \tag{7.39}$$

The probability, obtained at the last step of the procedure is the probability of satisfying the required quantity for all types of items for the next assembly shop (in a more general case, customer demands). The sizes of lots can be found by backtracking.

7.3.4.2 Genetic Algorithm

Coding of Solutions

The proposed GA searches for the optimal lot sizes. Taking into account the objective function, a solution (chromosome) is presented as follows: if the initial problem has n different products, then the chromosome will be an array of integer numbers of the length $n-1$. Each gene of chromosome is the size of appropriate lot (see Figure 7.9).

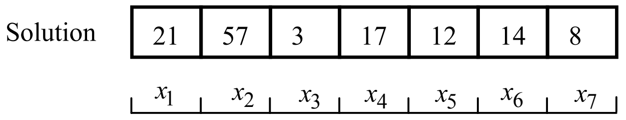


Fig. 7.9 An example of a chromosome for the problem of size 8

Fitness Function

The Fitness function is the probability to satisfy all demands, so it should be calculated as shown in the previous sections. There is an infinite sum in Eq. 7.30. Using the approximate method from Dolgui (2002), we can determine the upper and lower bounds of the value $p(T_{ps}^{act} \le T_{ps} \mid T^t)$:

$$\overline{p(T_{ps}^{act} \le T_{ps} \mid T^t)} = \exp\left\{-\left(T^tU + T_{ps}R\right)\right\} \times \sum_{v=0}^{H}\left(\left(\left(T_{ps}R\right)^v \Big/ v!\right)\sum_{j=0}^{v}\left(T^tU\right)^j \Big/ j!\right), \tag{7.40}$$

$$\underline{p(T_{ps}^{act} \le T_{ps} \mid T^t)} = 1 - \exp\left\{-\left(T^tU + T_{ps}R\right)\right\} \times \sum_{v=1}^{H}\left(\left(\left(T^tU\right)^v \Big/ v!\right)\sum_{j=0}^{v-1}\left(T_{ps}R\right)^j \Big/ j!\right). \tag{7.41}$$

The value H is the smallest integer for which the following equation is satisfied:

$$\overline{p(T_{ps}^{act} \le T_{ps} \mid T^t)} - \underline{p(T_{ps}^{act} \le T_{ps} \mid T^t)} \le 2\varepsilon\ , \tag{7.42}$$

where ε is a sufficiently small value. Then:

$$p(T_{ps}^{act} \le T_{ps} \mid T^t) = \left(\overline{p(T_{ps}^{act} \le T_{ps} \mid T^t)} + \underline{p(T_{ps}^{act} \le T_{ps} \mid T^t)} \right) \Big/ 2. \tag{7.43}$$

Thus, the fitness value $f(g)$ is assigned to each chromosome g as follows:

$$f(g) = \prod_{i=1}^{n-1} p_i^{>}\left(d_i \mid g_i\right) \times p_n^{>}\left(d_n \mid x_n^{+}, T^{+}(g), T_{nr}(g)\right). \tag{7.44}$$

Initial Population

At the beginning of the procedure we should create a certain quantity of solutions, i.e. an initial population. This population is generated randomly, simply considering the minimal and maximal possible sizes for each lot, i.e., each gene is generated respecting inequality $\underline{x}_i \le g_i \le x_i^{+}$, $i = 1,...,n-1$. Moreover, for each generated solution the following condition should be satisfied:

$$T_0 - S\left(\pi^*\right) - (k-1)\sum_{j=1}^{n} t_j - \sum_{j=1}^{n-1} t_j g_j - t_n d_n > 0 \tag{7.45}$$

Once the necessary quantity of solutions is obtained, then the uniqueness of each of them is checked. If there are clones in the population, then they are replaced by newly generated solutions.

Selection and Crossover

The *elitism strategy* was chosen to select individuals for crossover operation. First, 60% of the chromosomes are chosen from the current population to provide offspring. The standard two-point crossover is applied (see Figure 7.10).

A pair of parents is chosen randomly. Let c_1 and c_2, $0 \le c_1 < c_2 < n-1$ are the points of the crossover for the pair of parents $g^{p1} = (g_1^{p1},...,g_{n-1}^{p1})$ and $g^{p2} = (g_1^{p2},...,g_{n-1}^{p2})$, then the offspring g^{of1} and g^{of2} are as follows:

$$\begin{cases} g_i^{ofk} = g_i^{pk}, \text{ for } i \in \left[0,...,c_1 - 1\right] \quad \text{and} \quad k \in \{1,2\}, \\ g_i^{of1} = g_i^{p2}, \text{ for } i \in \left[c_1,...,c_2\right], \\ g_i^{of2} = g_i^{p1}, \text{ for } i \in \left[c_1,...,c_2\right], \\ g_i^{ofk} = g_i^{pk}, \text{ for } i \in \left[c_2 + 1,...,n-1\right] \quad \text{and} \quad k \in \{1,2\}. \end{cases} \tag{7.46}$$

If $c_1 = 0$, we deal with one-point crossover.

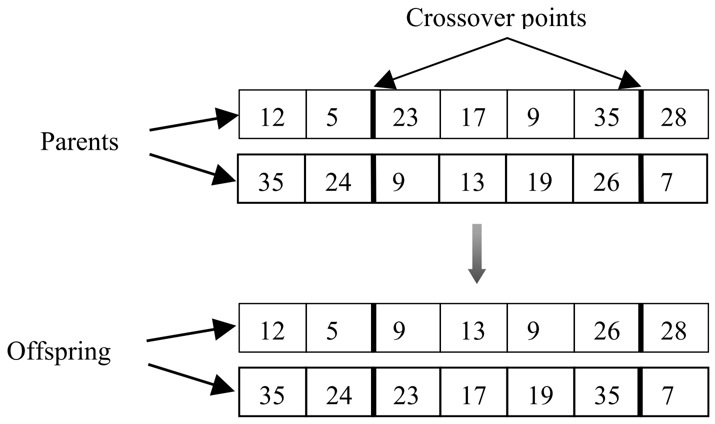


Fig. 7.10 An example of crossover operation for a problem of size 8

Mutation

The operation of mutation is intended to provide a wider range of possible solutions. The mutation is applied to offspring with a certain small probability. If the mutation will be applied to the offspring g^{of}, then the following steps are executed:

- First of all, generate randomly a number m in the range $[0, n-1]$. It is the position of a gene in the chromosome g^{of} that will be changed.
- Then, generate a number t from the set $\{-2,-1,1,2\}$. It is the value that we will add to the value of gene g_m^{of}.
- Finally, modify the gene: $g_m^{of_{new}} = g_m^{of} + t$.

As there are upper and lower limits for the lot sizes ($\underline{x}_i$ and x_i^+, $i = 1,...,n$), we should verify if the following condition is satisfied: $\underline{x}_m \le g_m^{of_{new}} \le x_m^+$, and, if it is necessary, provide the following operation:

$$\begin{cases} g_m^{of_{new}} = x_m^+, & \text{if} \quad g_m^{of_{new}} > x_m^+, \\ g_m^{of_{new}} = \underline{x}_m, & \text{if} \quad g_m^{of_{new}} < \underline{x}_m. \end{cases} \tag{7.47}$$

Local Search

Usually a local search procedure is applied to ameliorate the best solution of the current generation. For example, if the solutions $x^1, x^2, ..., x^r$ of a generation are sorted in a decreasing order of the fitness value $f(x^i)$, where r is the size of the population, then the local search is applied to the solution $x^1 = \left(x_1^1, x_2^1, ..., x_{n-1}^1\right)$. The procedure is as follows:

- For each $i, \quad i = 1,...,n-1$:
  - If $x_i^1 > \underline{x}_i$, calculate the fitness of chromosome $(x_1^1, x_2^1, ..., x_i^1 - 1, ..., x_{n-1}^1)$;
  - If $x_i^1 < x_i^+$, calculate the fitness of chromosome $(x_1^1, x_2^1, ..., x_i^1 + 1, ..., x_{n-1}^1)$.
- If some solutions with fitness value greater than $f(x^1)$ are found, then choose the best one to replace the solution x^1.

7.3.5 Numerical Example

In this subsection, we provide a numerical example. Here, the considered production line consists of three machines (*k* = 3) and five different types of products (*n* = 5). The time horizon is equal to 24 hours ($T_0 = 24$). All known data is given below.

Table 7.1 contains the customer demands for each type of product d_i, unit processing times t_i for each machine and minimal required service level β_i for each $i, \quad i = 1,...,5$.

Table 7.1 Input data for the problem with five types of products

Type of item	Demand d_i, #parts	Unit processing time t_i, hours	Minimal service level, %
1	42	0.058	0.93
2	29	0.167	0.92
3	53	0.049	0.94
4	49	0.065	0.91
5	27	0.136	0.93

Table 7.2 presents the MTTF and MTTR for each machine.

Table 7.2 Mean time to failure $1/u_q$ and mean time to repair $1/r_q$ (in hours) for each machine

Machine	MTTF	MTTR
1	352.1	0.668
2	483.3	0.973
3	384.7	0.354

Table 7.3 contains the reject probabilities for all types of items.

Table 7.3 Probabilities p_{iq} that an item of type i after processing on the machine q is of a good quality

Type of item	Machine		
	1	2	3
1	0.96	0.98	0.94
2	0.94	0.97	0.93
3	0.97	0.95	0.96
4	0.98	0.99	0.97
5	0.95	0.96	0.94

In Table 7.4, the set-up times are reported. Values $s_{i,0}$, $i=1,...,5$ in the first column are the set-up times for the product i, $i=1,...,5$ if it is the first on the line. The set-up time s_{ij} is the time which appears if the item(s) of type j is processed after the item of type i, $i=1,...,5$, $j=1,...,5$.

Table 7.4 Set-up times (in hours)

Type of item	1	2	3	4	5
Set-up for first lot	0.4	0.41	0.35	0.27	0.3
1	0	0.38	0.42	0.32	0.29
2	0.25	0	0.4	0.24	0.37
3	0.32	0.25	0	0.37	0.31
4	0.39	0.29	0.42	0	0.21
5	0.25	0.26	0.41	0.31	0

According to the decomposition approach proposed, we will resolve two subproblems separately. It is not too difficult to verify that the optimal sequence of lots for this example is as follows: $\pi^* = (i_1 = 3,\ i_2 = 2,\ i_3 = 4,\ i_4 = 5,\ i_5 = 1)$. The total set-up time is equal to $S(\pi) = s_{0,3} + s_{3,2} + s_{2,4} + s_{4,5} + s_{5,1} = 1.3$ hours (see Table 7.4).

The probability p_i that a given item of type i is of good quality is calculated according the formula (7.20) while using the data of Table 7.3. The results are as follows: $p_1 = 0.884352$, $p_2 = 0.847974$, $p_3 = 0.88464$, $p_4 = 0.941094$, $p_5 = 0.85728$. Considering Eqs. 7.23 and 7.24 and Table 7.2, we obtain the values U (MTTF) and R (MTTR), respectively: $U = 0.00750864$ and $R = 1.55438$.

The total loading time $(k-1)\sum_{i=1}^{n-1} t_i$ is equal to 0.95 (The data are in Table 7.1).

The values $\underline{x}_i$ and x_i^+, $i=1,...,5$ (see Table 7.5) are calculated using Eqs. 7.22, 7.31 and 7.32 and values β_i from Table 7.1.

Using Table 7.1, in Table 7.5 are reported the optimal sequence π^* of lots and corresponding values $\underline{x}_i$, x_i^+, p_i, $i=1,...,5$.

Table 7.5 Input data with limits for the lot sizes and reject probabilities

Lot number	i_j	Demand d_{i_j}, parts	Unit processing time t_{i_j}, hours	Minimal service level, %	$\underline{x}_{i_j}$	$x_{i_j}^+$	p_{i_j}
1	3	53	0.049	0.94	65	71	0.88464
2	2	29	0.167	0.92	38	44	0.847974
2	4	49	0.065	0.91	55	60	0.941094
3	5	27	0.136	0.93	35	41	0.85728
5	1	42	0.058	0.93	51	58	0.884352

Values $\underline{w}_{i_j}, \overline{w}_{i_j}$ and $\underline{V}_{i_j}, \overline{V}_{i_j}$ used for the dynamic programming approach (see Sect. 7.3.4.1) are presented in Table 7.6. The value T_1 is equal to:

$24 - 1.3 - 0.95 = 21.75.$

Table 7.6 Calculations for dynamic programming

Lot number, r	i_j	$\underline{w}_{i_j}$	$\overline{w}_{i_j}$	$\underline{V}_{i_j}$	$T_1 - \sum_{j=r+1}^{5} \underline{w}_{i_j}$	$\sum_{j=1}^{r} \overline{w}_{i_j}$	$\overline{V}_{i_j}$
1	3	3.185	3.479	3.185	4.111	3.479	3.479
2	2	6.346	7.348	9.531	10.457	10.872	10.457
3	4	3.575	3.9	13.106	14.032	14.727	14.032
4	5	4.76	5.576	17.866	18.792	20.303	18.792
5	1	2.958	3.364	—	—	—	—

The iterative procedure is the following. The first step is:

$$V^1 \in \left[\underline{V}_1, \overline{V}_1\right] = \left[\underline{w}_{\pi_1}, \overline{w}_{\pi_1}\right] = \left[\underline{x}_{\pi_1} t_{\pi_1}, x_{\pi_1}^+ t_{\pi_1}\right]$$

$$= \{3.185,\quad 3.234,\quad 3.283,\quad 3.332,\quad 3.381,\quad 3.43,\quad 3.479\},$$

so there are seven possible values of V^1. After calculating H_1 for each V_j^1, $j = 1,...,7$ from V^1 we obtain (see Table 7.7):

Table 7.7 Calculations of H_1

V^1	3.185	3.234	3.283	3.332	3.381	3.43	3.479
H_1	0.967047	0.982923	0.991558	0.996007	0.998188	0.999029	0.999667

Not all the calculation results are provided, because in the second level we have $\left|V^2\right| = 36$ possible values, in the third $\left|V^3\right| = 176$ and in the fourth $\left|V^4\right| = 517$.

After computing all values $H_i(V^i)$, $i = 2,3,4$ by Eq. 7.38, we apply Eq. 7.39.

For this numerical example the following solution was obtained:

$$\begin{pmatrix} \pi^* \\ x^* \end{pmatrix} = \begin{pmatrix} 3 & 2 & 4 & 5 & 1 \\ 67 & 39 & 57 & 36 & 57 \end{pmatrix}$$

with probability of demand satisfaction equal to 0.87413921.

We performed 10 program runs for this instance. In all runs the GA gave the exact solutions. For a run, the set of initial solutions with fitness values are presented in Table 7.8.

Table 7.8 Initial population for an instance with five lots

3	2	4	5	Fitness value
57	36	66	40	0.8524158
56	36	70	39	0.8492555
56	37	65	38	0.8373124
56	37	67	40	0.8267541
55	37	65	40	0.8245764

Fifty iterations (generations of the GA) were provided. In most runs the exact solution was obtained on the third generation.

In addition, a simulation model was developed with ARENA, and simulations were made using the Monte-Carlo approach. So, after a large number of simulations, we can estimate the probability of satisfying all demands. Figure 7.11 presents the average value of this estimation as a function of the number of simulations. A quite rapid convergence on the average is observed. After providing about 40,000 replications the quality of this average value is satisfactory.

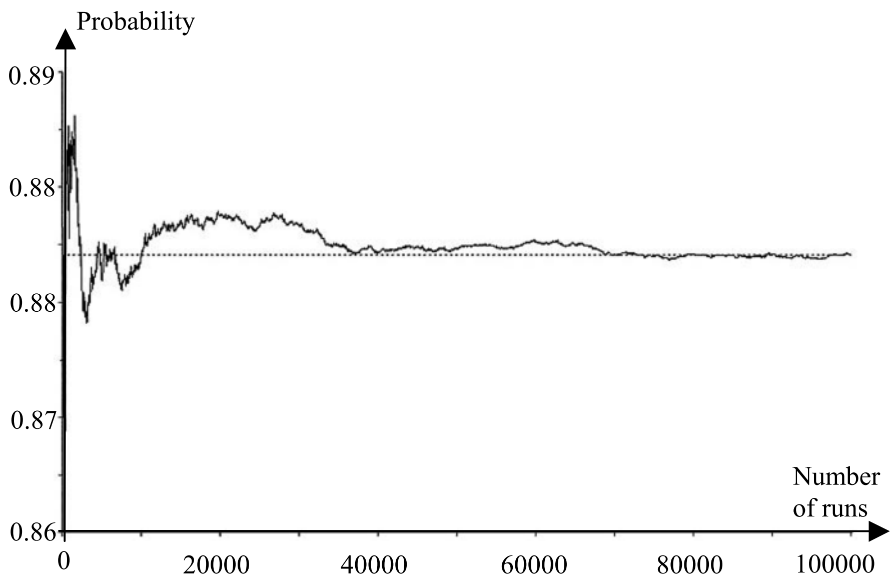


Fig. 7.11 The average value of estimation of the probability (using ARENA)

Note: this simulation model was used here for validation of the analytical results. Nevertheless, it is particularly useful for the further possible extensions of the proposed model, for example, if certain assumptions of the problem are relaxed, such as, fixed machine processing times or demands. Therefore, coupling simulation with optimization should be used for similar problems with both random demands and processing times.

7.3.6 Numerical Results

Some tests were effectuated to compare the two algorithms proposed (dynamic programming and genetic). Here, we provide the results. If we limit the calculation time to 1 hour, then the dynamic programming method can resolve the problems with number of lots up to and including 8. In Table 7.9, the computing times (average of 100 problem instances) of two algorithms are presented.

Table 7.9 Run time of exact and genetic methods (sec.) for $n \in [4,8]$. CPU time cannot exceed 3600 sec

Number of lots	4	5	6	7	8
Exact method (DP)	3.7	98.4	449.8	874.4	1590.2
GA	6.3	26.6	24.9	27.79	38.06

As shown, the CPU time for the dynamic programming (DP) approach rises very quickly, whereas the GA calculation time is significantly lower. For small problems, we verified also the average quality of approximate solutions obtained with the GA (see Table 7.10). The tests were made for 100 instances for each size of problem. "Relative error" is the mean of 100 values calculated as follows:

$$\frac{approximate\ solution\ \text{-}\ exact\ solution}{exact\ solution} \times 100\%\,.$$

Table 7.10 Relative error (in %) for approximate solutions delivered by GA for $n \in [4,8]$

Number of lots	4	5	6	7	8
Relative error, %	0.00364	0.02894	0.00421	0.00006	0.00001
Performance, %	97	90	90	98	99

In the row "Performance" of Table 7.10 the percentage of runs where the GA found an optimal solution is shown. We can see that the genetic approach gives quite good results for problems tested.

To estimate the quality of solutions given by the GA for the large-scale problems we use a simple heuristic algorithm:

1. First of all, create the solution vector $x^h = \left(x_1^h, x_2^h, \ldots, x_n^+\right)$, where $x_i^h = \left\lfloor \left(\underline{x}_i + x_i^+\right)/2 \right\rfloor$, $h = 1, \ldots, n-1$. Remember that $\lfloor a \rfloor$ is the largest integer number which does not exceed a;
2. Calculate the value $NumSteps = \max\limits_{i=1,\ldots,n} \left\{x_i^+ - \underline{x}_i\right\}$
3. To ameliorate the solution obtained in step 1, apply a local search procedure (see Sect. 7.3.4.2) $NumSteps$ times.

In Table 7.11, we compare the GA with this heuristic algorithm. Tested instances of the problem have the sizes from 10 to 100.

The CPU time of heuristic approach is less than that of the GA. However, the quality of the GA is much better. Moreover, this gap increases with increasing the number of lots.

Table 7.11 Comparison GA with heuristic algorithm

Number of lots	Genetic algorithm	Heuristic algorithm	
	CPU time, sec	CPU time, sec	Deviation,%
10	46.92	1.343	4.6
20	61.75	3.016	8.88
30	77.86	6.032	13.92

Table 7.11 (Continued)

Number of lots	Genetic algorithm	Heuristic algorithm	
	CPU time, sec	CPU time, sec	Deviation,%
40	162.01	9.344	19.84
50	477.3	11.218	25.82
60	624.32	25.075	31.32
70	783.69	60.834	37.67
80	977.1	82.7418	41.72
90	1122.38	98.8	57.76
100	1338.38	120.68	59.89

7.4 Conclusions

Lot-sizing and sequencing techniques help to make optimal production planning decisions. They provide answers to the following questions:

- What products should be chosen for processing?
- How many units of each product are necessary to launch on input of the production system to satisfy customer demands on output?
- In what order should the lots (products) be processed?
- Which machines should be used for processing a product?

The problems of lot-sizing and scheduling can have numerous objectives: to minimize the total cost, limit time of production, minimize the cost of delays (generally cost of the delay is paid for each backlogged item), and maximize the probability of satisfying the customer demands, etc.

In this chapter, some standard lot-sizing models and their extensions were presented and discussed. We showed that the majority of lot-sizing and scheduling problems is NP-hard.

The techniques applied for resolving lot-sizing and sequencing problems were also discussed. All these methods may be divided into three main groups: exact, ε -approximate and heuristics. Exact and ε -approximate methods give optimal or near optimal solutions, but the working time can be excessive. Heuristics have also a disadvantage: it is difficult to estimate the quality of the obtained solution.

The core of this chapter is a case study for lot-sizing and sequencing of flow-shop lines under uncertainties. This is a multi-product problem. Two main types of uncertainties were considered: breakdowns (random lead-time) and rejects (random yield). Lead-time is random because of machine breakdowns which appear at a certain breakdown rate; the repair durations are also random. The yield appears random because of the presence of rejects among the finished items. The objective was to maximize the probability that customer demands will be satisfied at the given due date. It was shown that it is possible to decompose the problem into lot-sizing and sequencing sub-problems. The later is a modification of the well-known travelling salesman problem. The former is a specific case of the knapsack problem. Two solution methods were proposed to resolve this particular knapsack problem. The first is a dynamic programming procedure, which searches for the optimal solutions. The second method is a GA. A comparative study of the two algorithms was made.

Further research might be focused on developing faster algorithms for both dynamic programming and genetic approaches. For more sophisticated problems, such as if the processing times are also random and the line is of more complex structure, we can combine the developed optimization methods and simulation model.

Acknowledgements The authors thank Chris Yukna for his help with English and Genrikh Levin for our discussions and his ideas on the previous step of this study.

References

Agnihothri S, Lee JS, Kim J (2000) Lot sizing with random yields and tardiness costs. Comput. Oper. Res. 27:437–459

Allahverdi A, Gupta JND, Aldowaisan T (1999) A review of scheduling research involving setup considerations. Int. J. Mgmt. Sci. 27:219–239

Allahverdi A, Ng CT, Cheng TCE et al. (2008) A survey of scheduling problems with setup times or costs. Eur. J. Oper. Res., 187:985–1032

Axsater S (2006) Inventory Control (2006) International series in operations research and management science. Springer Science + Business Media, LLC (second edition)

Cakanyildirim M, Bookbinder JH, Gerchak Y (2000) Continuous review inventory models where random lead-time depends on lot-size and reserved capacity. Int. J. Prod. Econ., 68: 217–228

Cardenas-Barron LE (2001) The economic production quantity (EPQ) with shortage derived algebraically. Int. J. Prod. Econ. 70:289–292

Chang SKJ, Chuang GPC, Chen HJ (2005) Short comments on technical note—The EOQ and EPQ models with shortages derived without derivatives. Int. J. Prod. Econ. 97:241–243

Chatfield DC (2007) The economic lot-scheduling problem: A pure genetic search approach. Comput. Oper. Res., 34:2865–2881

Dolgui A (2002) Performance analysis model for systems described by renewal process. Eng. Simul. (Electron. Model.) 24(2):3–11

Dolgui A, Hnaien F, Louly MA et al. (2008a) Parameterization of MRP for supply planning under lead-time uncertainties. In: Kordic, V. (Ed.) Supply Chain, Theory and Applications, pp. 247-262. I-Tech Education and Publishing, Austria

Dolgui A, Kovalyov MY, Shchamialiova K (2008b) Multi-product lot-sizing and sequencing on a single imperfect machine. In Le Thi HA, Bouvry P and Pham Dinh T (Eds) Modelling, Computation and Optimization in Information Systems and Management Sciences (MCO'2008), Communications in Computer and Information Science, 14:117-125. Springer-Verlag

Dolgui A, Levin G, Louly MA (2005) Decomposition approach for a problem of lot-sizing and sequencing under uncertainties. Int. J. Comput. Integr. Manuf., 18(5):376–385

Dolgui A, Louly MA (2002) A model for supply planning under lead-time uncertainty. Int. J. Prod. Econ., 78:145–152

Dolgui A, Prodhon C (2007) Supply planning under uncertainties in MRP environments: A state of art. Annu. Rev. Control, 31:269–279

Drexl A, Kimms A (1997) Lot-sizing and scheduling — Survey and extensions. Eur. J. Oper. Res., 99:221–235

Emmons H, Mathur K (1995) Lot-sizing in a no wait flow-shop. Oper. Res. Lett. 17:159–164

Eroglu A, Ozdemir G (2007) An economic order quantity model with defective items and shortages. Int. J. Prod. Econ. 106:544–549

Febbraro AD, Minciardi R, Sacone S (2001) Single machine scheduling with fixed lot-sizes and variable processing times. In: Proceedings of 40th IEEE Conference on Decision and Control (CDC), pp. 2349–2354

Fong DKH, Gempesaw VM, Ord JK (2000) Analysis of a dual sourcing inventory model with normal unit demand and Erlang mixture lead times. Eur. J. Oper. Res., 120:97–107

Gaafar L (2006) Applying genetic algorithms to dynamic lot-sizing with batch ordering. Comput. Ind. Eng., 51:433–444

Gerchak Y, Parlar M (1986) A single-period production model with uncertain output and demand. In: Proceedings of 25th IEEE Conference on Decision and Control (CDC), pp. 1733–1736

Gerchak Y, Wang Y, Yano CA (1994) Lot sizing in assembly systems with random yields. IIE Trans., 26(2):19–24

Giard V (2003) Gestion de la production et des flux. Production et techniques quantitatives appliquées à la gestion. Ed. Economica

Grosfeld-Nir A, Gerchak Y (1996) Production to order with random yields: single stage multiple lot-sizing. IIE Trans., 28:669–676

Grosfeld-Nir A, Gerchak Y (2004) Multiple lot-sizing in production to order with random yields: Review of recent advances. Ann. Oper. Res., 126:43–69

Grubbstrom RV, Erdem A (1999) The EOQ with backlogging derived without derivatives. Int. J. Prod. Econ. 59:529–530

Gurnani H, Gerchak Y (2007) Coordination in decentralized assembly systems with uncertain component yields. Eur. J. Oper. Res., 176:1559–1576

Guu SM, Liou FR (1999) An algorithm for the multiple lot-sizing problem with rigid demand and interrupted geometric yield. J. Math. Anal. Appl., 234:567–579

Guu SM, Zhang AX (2003) The finite multiple lot sizing problem with interrupted geometric yield and holding costs. Eur. J. Oper. Res., 145:635–644

Haji R, Haji A, Sajadifar M, Zolfaghari S (2008) Lot sizing with non-zero setup times for rework. J. Syst. Sci. Syst. Eng., 17(2):230–240

Hernandez W, Suer GA (1999) Genetic algorithms in lot sizing decisions. In: Proceedings of the 1999 Congress on Evolutionary Computation (CEC99), 3:2280–2286

Inderfurth K (2009) How to protect against demand and yield risks in MRP systems. Int. J Prod. Econ., DOI: 10.1016/j.ijpe.2007.02.005

Inderfurth K, Janiak A, Kovalyov MY et al. (2006) Batching work and rework processes with limited deterioration of reworkables. Comput. Oper. Res., 33:1595–1605

Kacem I, Mahjoub AR (2009) Fully polynomial time approximation scheme for the weighted flow-time minimization on a single machine with a fixed non-availability interval. Comput. and Ind. Eng., DOI:10.1016/j.cie.2008.09.042

Karimi B, Fatemi Ghomi SMT, Wilson JM (2003) The capacitated lot sizing problem: a review of models and algorithms. Omega, 31:365–378

Khouja M, Goyal S (2005) Single item optimal lot sizing under continuous unit cost decrease. Int. J. Prod. Econ., 102:87–94

Kis T, Pesch E (2005) A review of exact solution methods for the non-preemptive multiprocessor flowshop problem. Eur. J. Oper. Res., 164:592–608

Kubiak W, Cheng J, Kovalyov MY (2002) Fast fully polynomial approximation schemes for minimizing completion time variance. Eur. J. of Oper. Res., 137:303–309

Lee H, Billington C (1993) Material management in decentralized supply chains. Oper. Res., 41: 837–845

Lee I, Sikora R, Shaw MJ (1997) A genetic algorithm-based approach to flexible flow-line scheduling with variable lot sizes. IEEE Trans. Syst. Man Cybernetics, 27:36–54

Li Q, Xu H, Zheng S (2008) Periodic-review inventory systems with random yield and demands: Bounds and heuristics. IIE Trans., 40:424–444

Liao GL, Chen YH, Sheu SH (2009) Optimal economic production quantity policy for imperfect process with imperfect repair and maintenance. Eur. J. Oper. Res., 195:348–357

Louly MA, Dolgui A (2004) The MPS parameterization under lead-time uncertainty. Int. J. Prod. Econ. 90:369–376

Louly MA, Dolgui A (2009) Calculating safety stocks for assembly systems with random component procurement lead times: A branch and bound algorithm. Eur. J. Oper. Res., 199:723–731

Maddah B, Jaber M Y (2008) Economic order quantity for items with imperfect quality: Revisited. Int. J. Prod. Econ., 112:808–815

Minner S (2007) A note on how to compute economic order quantities without derivatives by cost comparisons. Int. J. Prod. Econ., 105:293–296

Molinder A (1997) Joint optimization of lot-sizes, safety stocks and safety lead times in an MRP system. Int. J. Prod. Res., 35(4):983–994

Pochet Y, Wolsey LA (2006) Production planning by mixed integer programming. In Mikosch TV, Resnick SI, Robinson SM (eds.). Springer series in operations research and financial engineering. Springer Science + Business Media

Potts CN, Kovalyov MY (2000) Scheduling with batching: A review. Eur. J. Oper. Res., 120: 228–249

Radhoui M, Rezg N, Chelbi A (2009) Integrated model of preventive maintenance, quality control and buffer sizing for unreliable and imperfect production systems. Int. J. Prod. Res., 47(2): 389–402

Richter K, Pakhomova NV, Dobos I (2006) A Wagner/Whitin natural resource stock control model. Int. J. Prod. Econ., 104:419–426

Ronald R, Yang GK, Chu P (2004) Technical note: The EOQ and EPQ models with shortages derived without derivatives. Int. J. Prod. Econ., 92:197–200

Sikora R (1996) A genetic algorithm for integrating lot-sizing and sequencing in scheduling a capacitated flow line. Comput. Ind. Eng., 30(4):969–981

Singh MR, Abraham CT, Akella R (1998) Planning production of a set of components when yield is random. In: Electronic Manufacturing Technology Symposium, 1988, 'Design-to-Manufacturing Transfer Cycle'. Fifth IEEE/CHMT International, USA, pp. 196–200

Tajbakhsh MM, Zolfaghari S, Lee CG (2007) Supply uncertainty and diversification: a review. In Jung H, Jeong B, Chen FF (eds) Trends in supply chain design and management, technologies and methodologies part 2, pp. 345–368, Springer, London

Tempelmeier H, Buschkuhl L (2008) Dynamic multi-machine lotsizing and sequencing with simultaneous scheduling of a common setup resource. Int. J. Prod. Econ.,113: 401–412

Teng JT (2009) A simple method to compute economic order quantities. Eur. J. of Oper. Res., 198:351–353

Teunter RH, Flapper SDP (2003) Lot-sizing for a single-stage single product production system with rework of perishable production defectives. OR Spectrum 25:85–96

Teunter RH, Tang O, Kaparis K (2009) Heuristics for the economic lot-scheduling problem with returns. Int. J. Prod. Econ., 118:323–330

Wang Y, Gerchak Y (2000) Input control in a batch production system with lead times, due dates and random yields. Eur. J. Oper. Res., 126:371–385

Wilhelm WE, Som P (1998) Analysis of a single-stage, single product, stochastic, MRP-controlled assembly system. Eur. J. Oper. Res., 108:74–93

Yano CA, Lee HL (1995) Lot sizing with random yields: A review. Oper. Res., 43(2):311–333
Yao DD (1988) Optimal run quantities for an assembly system with random yields, IIE Trans., 20(4):399–403
Yao MJ, Elmaghraby SE (2001) The economic lot scheduling problem under power-of-two policy. Comput. Math. Appl., 41:1379–1393
Yao MJ, Huang JX (2005) Solving the economic lot scheduling problem with deteriorating items using genetic algorithms. J. Food Eng., 70:309–322
Zhang AX, Guu SM (1997) Properties of the multiple lot-sizing problem with rigid demands and general yield distributions. Comput. Math. Appl., 33(5):55–65
Zhu X, Wilhelm WE (2006) Scheduling and lot-sizing with sequence-dependent setup: A literature review. IIE Trans., 38:987–1007

Chapter 8
Meta-heuristics for Real-time Routing Selection in Flexible Manufacturing Systems

M. Souier, A. Hassam and Z. Sari

Abstract Most studies in real-time flexible manufacturing system (FMS) scheduling and control areas do not consider the effect of routing flexibility; their focus is typically on use of scheduling (i.e., dispatching) rules based on routing selection carried out prior to production. Such an approach is not applicable to random-type FMS, in which no knowledge about incoming part types is available prior to production. For such a scenario, parts can have alternative routings, even for parts of the same type. Thus, the control system of a random-type FMS requires the capability to adapt to the randomness in arrivals and other unexpected events in the system by effectively using operation and routing flexibility in real time. In this chapter, the objective is to present a comparative study of a group of meta-heuristics, including taboo search, ant colony optimization, genetic algorithms, particle swarm optimization, electromagnetic meta-heuristic, and simulated annealing, against the modified dissimilarity maximization method (modified DMM). DMM (Saygin and Kilic, 1999) is an alternative process plan selection method originally proposed for the routing selection in off-line scheduling of an FMS. In subsequent studies (Saygin *et al.*, 2001; Saygin and Kilic, 2004) DMM has been: (i) used as a real-time decision-making tool to select routings for the parts that are in the system, (ii) tested and benchmarked against first-in-first-out/first available and equal probability loading. Based on the DMM model, a modified DMM (Hassam and Sari, 2007) is developed for selection of alternative routings in real time in an FMS. Modified DMM improves the performance of the FMS in terms of higher production rate, and higher utilization rate of the machines and the material handling system.

Keywords Meta-heuristic, real-time routing, flexible manufacturing system, job-shop, scheduling

M. Souier, A. Hassam and Z. Sari (✉)
LAT, Aboubekr Belkaïd University of Tlemcen, PoBox 230, Tlemcen 13000, Algeria
e-mail: z.sari@univ-tlemcen.dz

8.1 Introduction

Nowadays, businesses are facing increased competition; pressure for higher variety of customized products, shorter lead times, higher quality, and lower cost due to competitors. Today, having flexibility in production and effectively managing it to be more competitive is more crucial than ever. Several decades after their design, flexible manufacturing systems (FMS) still provide great flexibility. They provide various benefits, such as high resource utilization, high productivity, reduced work-in-process, and many more. In such systems, resource allocation decisions, process planning, and scheduling of operations are generally made dynamically and in a very short time almost in real time, depending on the state of the production system (availability of resources, availability of material handling, presence of bottlenecks), the characteristics of the production plan (due date of manufacturing orders) and the production targets (production rate increase, work-in-process reduction).

The real-time scheduling of operations uses multiple approaches, such as selection of parts in buffers for immediate machining and selection of machines for a part among alternative machines by priority (i.e., scheduling or dispatching) rules, which is one of the simplest and most commonly used methods. These priority rules have been studied for many years. Saygin and Kilic (1999) presented an extensive literature survey.

Among the rules and methods of scheduling in real time, we can find the dissimilarity maximization method (DMM) (Saygin *et al.*, 2001, Saygin and Kilic, 2004) and the modified DMM (Hassam and Sari, 2007). The former is a rule for selecting alternative routing in real time in an FMS based on its original version for off-line routing selection (Saygin and Kilic, 1999). The latter is an improvement of the DMM rule in order to improve the performance of the production system. These methods use coefficients of dissimilarity between the machines to make decisions related to routings. The scheduling problems in manufacturing systems are generally NP hard and there are not universal methods making it possible to solve effectively all the cases (Garey and Johson, 1979).

Meta-heuristics are algorithms of the stochastic type aiming to solve a broad range of hard optimization problems, for which one does not know more effective traditional methods, often inspired by analogies with reality, such as physics (simulated annealing, simulated diffusion), biology (evolutionary algorithms, taboo search), and ethnology (ant colony, swarm intelligence). They are generally of discrete origin, but can be adapted to the other types of problems and they share also the same disadvantages: difficulties of parameter adjustment and large computation time.

In this chapter, our interest is focused on a group of meta-heuristics, which include in particular simulated annealing (SA), genetic algorithm (GA), taboo search (TS), ant colony algorithms (ACO), particle swarm optimization (PSO) and electromagnetism-like method (EM). We present a comparative study between these meta-heuristics and the modified DMM rule. We will see that the meta-heuristics are largely based on a common set of principles, which make it possible to design solution algorithms; the various regroupings of these principles thus lead to a large variety of meta-heuristics.

The rest of the chapter is organized as follows. Sect. 8.2 presents our literature review from scheduling problems and rules, routing selection methods and meta-heuristics and their applications in routing selection. Sect. 8.3 discusses the problem under consideration. Sect. 8.4 presents more in detail the original DMM and modified DMM. Sect. 8.5 illustrates the different meta-heuristics for job-shop routing where six meta-heuristics, respectively ACO, SA, PSO, GA, TS and EM are presented. Sect. 8.6 shows the performance evaluation for the routing selection method. Sect. 8.7 concludes the chapter with some directions for future work.

8.2 Literature Review

The scheduling problems are usually NP hard. One of the first studies of the scheduling of FMS is the work of Nof *et al.* (1979), where they demonstrate the importance and effect of scheduling decisions on various performance measures of production systems. Traditional scheduling involves sequencing of operations and time allocation of their start and end times before the production starts. Traditional scheduling requires that production orders in terms of part types and their routings are known prior to production. On the other hand, real-time scheduling is carried out as a control activity, which involves real-time decision-making in terms of selection of part types and their routings as parts come in to the production system. This category of scheduling problems involves various challenges, such as variable part arrival rates, unexpected breakdowns, need for synchronization of tool management, effective management of material handling systems, and lack of raw materials.

The factors listed above, and many others, make reordering required to avoid the increase in waiting time, the increase in work-in-process, the low utilization of machinery and equipment and possibly the degradation of the production system performance (Wu and Wysk, 1989; Ishii and Muraki, 1996). Several researchers propose different methods to provide maximum flexibility in real-time scheduling in order to increase the performance of systems (Saygin and Kilic, 1999; Liu and MacCarthy, 1997, 1996, etc.). However, the real-time scheduling is always a desirable but elusive goal (Basnet and Mize, 1994; Shukla and Chen, 1996). Consequently, establishing an integrated system for real-time scheduling and control that responds to changes in the state of the system is essential to improve the performance of a production system.

The control and real-time scheduling of flexible production systems have become a popular research area since the early 1980s, a period in which flexible production systems were adopted by the industrialized countries (Saygin *et al.*, 1995; Saygin and Kilic, 1997; Peng and Chen, 1998). But many studies in controlling and scheduling of FMS in real time do not take into account the flexibility of alternative routing (Byrne and Chutima, 1997; Kazerooni *et al.*, 1997). However, most studies that take into account this point handle the problem of routing selection prior to the start of production (Das and Nagendra, 1997; Cho and Wysk, 1995).

Scheduling rules have been studied by many researchers. The common conclusion among these studies includes: first, the results are dependent on the production system that has been studied; therefore, they cannot be generalized. Second, these rules are myopic in nature; therefore they lead to imperfect scheduling since they do not capture the relevant information at various levels of production systems (Rachamadugu and Stecke, 1994; Gupta *et al.*, 1989; Kouiss *et al.*, 1997). The weakness of these scheduling rule-based approaches in handling real-time scheduling in FMS has been the major driving force behind the *development* of new methods for alternative routing selection in real time.

During the last few years, a new family of approximate algorithms and methods has dominated the combinatorial optimization problem solution research. These methods and algorithms basically combine heuristic methods in higher-level frameworks. The goal of this new methodology is to efficiently and effectively explore the search space driven by logical moves and knowledge of the effect of a move facilitating the escape from locally optimum solutions (Blum and Roli, 2003). These methods are nowadays called meta-heuristics, a term that was first introduced by Glover (Glover and Greenberg, 1989). Meta-heuristics have an advantage over simpler heuristics in terms of solution robustness. However, they are usually more difficult to implement and tune, as they need special information about the problem to be solved to obtain the best results. Due to the computational complexity of combinatorial optimization problems, the moderate results acquired by heuristic methods and the time limitations for application of exact algorithms, the application of meta-heuristic methods to solve scheduling problems is a well-established field of research.

In their original definition, meta-heuristics are methods that organize interaction between local improvement procedures and higher-level strategies for creating a process able to escape from local optima and perform a robust search of a solution space. Meta-heuristics have also come to include any procedures that employ strategies for overcoming the trap of local optimality in complex solution spaces, especially those procedures that utilize one or more neighbourhood structures as a means of defining admissible moves to transition from one solution to another, or to build or destroy solutions in constructive and destructive processes. Almost all meta-heuristic algorithms proposed in the literature have been employed to solve shop-scheduling problems (Blum and Roli, 2003). In this chapter we present six meta-heuristics for job-shop routing: ACO, SA, PSO, GA, TS and EM. These meta-heuristics are the most popular for solving the problems of job-shop scheduling. The results obtained by these meta-heuristics are compared to the modified DMM, which is a rule of selection of alternative routing in real time.

8.3 Job-shop

8.3.1 Job-shop Problem

The job-shop scheduling problem consists of a set of N jobs (1, 2, . . ., n) to be processed on a set of M machines (1, 2, . . ., m). In job-shop scheduling, each job must be processed on every machine and consists of a sequence of m_i operations, which have to be scheduled in a predetermined order, different for each job. Each job can be processed on only one machine at a time and each machine can process only one job at a time. The operation setup times are included in the processing times and are independent of sequencing decisions. The scheduling problem lies in finding a sequence of jobs for each machine that optimizes specific performance. Moreover, each job has its own predetermined route to follow. The simplest job-shop models assume that a job may be processed on a particular machine at most once on its route through the system (see Figure 8.1).

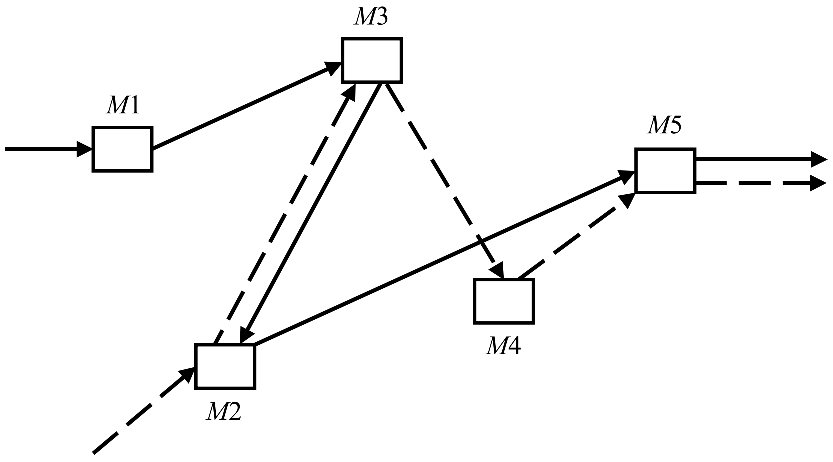


Fig. 8.1 A simple example of job-shop (M= machine; ——▸ , routing 1; ---▸ , routing 2)

In some others, a job may visit a machine several times on its route through the system. These shops are said to be subject to recirculation, which increases considerably the complexity of the model. The routes of the jobs are order-specific and require recirculation. More general models assume a production environment that consists of a network of interconnected facilities with each facility being a (flexible) flow-shop or a (flexible) job-shop. At a higher level, supply chain management also makes use of such planning and scheduling of networks for streamlining supply and demand among the business partners and end customers.

8.3.2 Simulation of a Flexible Manufacturing System Environment

In order to compare meta-heuristics and the modified DMM, we use a simulation model of an FMS environment. This FMS model was used by Saygin and Kilic (1999) and Hassam and Sari (2007); it contains seven machines, a loading station, an unloading station, and one automated guided vehicle (AGV). Six different types of parts are considered for production in the system. The machines and stations are as follows:

1. Two vertical milling machines (VMC).
2. Two horizontal milling machines (HMC).
3. Two vertical turning centres (VTC).
4. One shaper (SHP).
5. One loading station (L).
6. One unloading station (UL).

Each machine has an input buffer and an output buffer. The loading station also contains an input buffer. The configuration of the considered FMS is given in Figure 8.2.

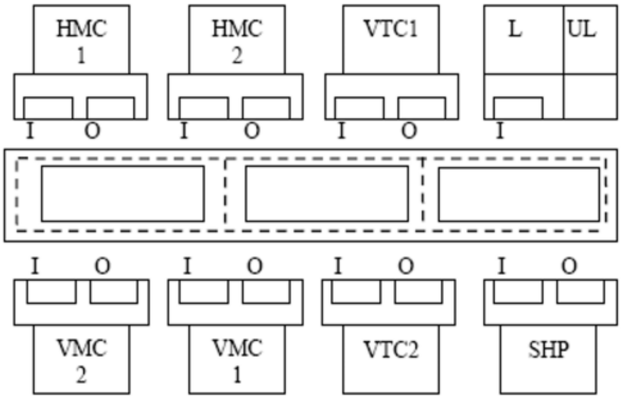


Fig. 8.2 Configuration of the FMS (*I*, input buffer; *O*, output buffer; …., AGV routes)

The studied operations on the flexible production system are based on the following assumptions:

- The alternative routings of each type of part are known before the start of production.
- The AGV routes depend on the selected alternative routings in real time.
- The processing time is known.

- The processing time of an operation is the same on the alternative machines identified for this operation.

Each machine can process one piece at a time. The alternative routing and the processing time for each type of part are given in Table 8.1.

Table 8.1 Alternative routings of part types

Part type and production ratio	Routings (processing time)
A (17%)	L – VTC1 (30) – VMC1 (20) - UL L – VTC1 (30) – VMC2 (20) - UL L – VTC2 (30) – VMC1 (20) - UL L – VTC2 (30) – VMC2 (20) – UL
B (17%)	L – VTC1 (20) – SHP (1) – VMC1 (15)-UL L – VTC1 (20) – SHP (1) – VMC2 (15)- UL L – VTC2 (20) – SHP (1) – VMC1 (15) - UL L – VTC2 (20) – SHP (1) – VMC2 (15) - UL
C (17%)	L – VTC1 (40) – VMC1 (25) - UL L – VTC1 (40) – VMC2 (25) - UL L – VTC2 (40) – VMC1 (25) - UL L – VTC2 (40) – VMC2 (25) – UL
D (21%)	L – VTC1 (40) – SHP (1) – VTC1 (20) – HMC1 (35)–UL L – VTC1 (40) – SHP (1) – VTC1 (20) – HMC2 (35)–UL L – VTC1 (40) – SHP (1) – VTC2 (20) – HMC1 (35)–UL L – VTC1 (40) – SHP (1) – VTC2 (20) – HMC2 (35)–UL L – VTC2 (40) – SHP (1) – VTC1 (20) – HMC1 (35)–UL L – VTC2 (40) – SHP (1) – VTC1 (20) – HMC2 (35)–UL L – VTC2 (40) – SHP (1) – VTC2 (20) – HMC1 (35)– UL L – VTC2 (40) – SHP (1) – VTC2 (20) – HMC2 (35)–UL
E (20%)	L – VTC1 (25) – SHP (1) – VTC1 (35) – HMC1 (50)–UL L – VTC1 (25) – SHP (1) – VTC1 (35) – HMC2 (50)–UL L – VTC1 (25) – SHP (1) – VTC2 (35) – HMC1 (50)–UL L – VTC1 (25) – SHP (1) – VTC2 (35) – HMC2 (50)–UL L – VTC2 (25) – SHP (1) – VTC1 (35) – HMC1 (50)–UL L – VTC2 (25) – SHP (1) – VTC1 (35) – HMC2 (50)–UL L – VTC2 (25) – SHP (1) – VTC2 (35) – HMC1 (50)–UL L – VTC2 (25) – SHP (1) – VTC2 (35) – HMC2 (50)–UL
F (8%)	L –HMC1 (40) – UL L –HMC2 (40) – UL

8.4 Dissimilarity Maximization Method and Modified DMM Rules

8.4.1 Dissimilarity Maximization Method for Real-time Routing Selection

Saygin and Kilic (1999) developed the original DMM. Inspired from group technology, DMM is used for selecting alternative process plans for the selection of alternative routing to schedule off-line FMS. It has a reciprocal function in group technology, as it tends to maximize the dissimilarity coefficients instead of similarity coefficients. This rule selects routings for the parts in the system among their alternative routings where the total dissimilarity among the selected routings is maximized. Dissimilarity between two routings is defined in terms of machines that belong to each routing. The selection of a routing among alternative routings of each part is performed according to the maximization of the sum of the dissimilarity coefficients. This rule was developed to reduce congestion and increase production rate in FMS. The following notation has been used to formalize the DMM method:

n:	number of parts;
q:	number of routings;
D_{ij} :	dissimilarity between routings i and j;

C_{ij}= 1 if routing j belongs to the routings of part i. Otherwise, $C_{ij} = 0$;
$X_j = 1$ if routing j is selected. Otherwise, $X_j = 0$;

S_j:	sum of maximum dissimilarity.

The dissimilarity coefficient (dissimilarity of machine type) between two routings i and j is defined as follows (Saygin and Kilic, 1999):

$$D_{ij} = \frac{\text{Number of machine types that are not common in both routing } i \text{ and } j}{\text{Total number of machine types in both routing}} . \quad (8.1)$$

For the selection of alternative routing one should maximize the total sum of dissimilarities between the routing as follows (Saygin and Kilic, 1999):

$$S_j = \text{Max} \sum_{i=1}^{q} \sum_{j=1}^{q} X_j D_{ij} \quad (8.2)$$

subject to:

$$\sum_{j=1}^{q} C_{ij} X_j = 1 \quad \text{for all parts } i = 1, \cdots, n , \quad (8.3)$$

$$\sum_{j=1}^{q} X_j = n \quad \text{for all routings } j = 1, \cdots, q . \quad (8.4)$$

Equation 8.3 states that only one routing will be selected for each part. Equation 8.4 states that the number of selected routings will be equal to the number of parts.

8.4.2 Modified Dissimilarity Maximization Method for Real-time Routing Selection

Based on the original DMM rule, Hassam and Sari (2007) proposed a modified version of DMM rule called the modified DMM rule. The major motivation behind the modified DMM was twofold. For high arrival rate of parts and small buffer capacities, the production system is overloaded and yet the utilization rates of the machines and the material handling system are low.

These two factors affect the performance of the FMS. Hence, we propose the modified DMM rule to overcome these problems. Our modification of DMM rule is intended to keep the same principle, which depends on the maximization of dissimilarity coefficients for the selection of various routing, but by affecting several parts to a single routing. So if all routes are selected by a part, the newly created part will choose among routings; the part will be delivered in the routing where the queue of the first machine of this routing contains at least one free place. The parts arriving first have a higher priority following the first-in-first-out rule; the other parts will wait in input or output queues of various machines or in the loading station. The modified rule will use the following algorithm for the selection of alternative routing in real time in a flexible production system.

Step 1: All routes are free (available) so $X(i) = 0$
Step 2: Calculation of dissimilarity coefficients $D_i(1)$
Step 3: Creation (arrival) of parts
Step 4: Condition: depending on the type of part tested:
If there's at least one free routing and at least one free place in the queue of the loading station
Or
If all routes are busy and the input queue of the first machine of at least one routing contains at least one free place and this machine is not broken down
Step 5: If the previous condition is not verified, the part is in a queue until the condition is verified
Step 6: If the condition of step 4 is checked then we calculate the sum:

$$S(j) = \sum_{i=1}^{q} X(i)D(i,j)\,. \tag{8.5}$$

Step 7: Test, if there is a maximum of $S(j)$ (there are free routings)
Step 8: If the previous condition is checked then go to step 10

Step 9: If the condition of step 7 is not checked, then select the routing where the input queue of its first machine contains at least one free place
Step 10: Routing j selected according to step 7 or step 9 is occupied, $X(j) = 1$
Step 11: Treatment of the part according to the selected routing j
Step 12: At the end of treatment, routing becomes available again, $X(j) = 0$
Step 13: Part leaves the system

The cycle repeats itself from step 3 to step 11 every time a part arrives.

8.5 Meta-heuristics for Job-shop Routing

8.5.1 Ant Colony Optimization

This meta-heuristic was introduced by Dorigo (1992) and was inspired by the studies on the real ant whose members are individually equipped with very limited faculties but can find the shortest path from a food source to their nest without visual cues. They are also capable of adapting to changes in the environment like the appearance of an unexpected obstacle on the initial path between the food source and the nest. The first algorithm of this type of meta-heuristics was designed to solve the travelling salesman problem (Dorigo, 1992). This algorithm principle is simple. The following notation is used:

k :	ant;
i, j:	city;
T:	iteration;
p^k_{ij}:	probability of following the trail ij;
$\tau^k_{ij}(t)$:	intensity of the trail ij;
η_{ij}:	visibility of the trail ij;
$\Delta\tau^k_{ij}(t)$:	the quantity of pheromone deposited by ant k;
$T^k(t)$:	the way effected by the ant k at t iterations.

When an ant k moves from city i to city j, it leaves a trail on the way. Moreover, it chooses the next city to be visited using a probability p^k_{ij} based on a compromise between the intensity of the trail $\tau^k_{ij}(t)$ and visibility η_{ij} that represents the reciprocal of the distance between i and j, the relative importance of the two elements is controlled by two parameters α and β.

Each ant k has a form of memory tabu$_k$; it points out the ordered list of the cities which have been already visited in order to force this one to form an acceptable solution. After a full run, each ant deposit a certain quantity of pheromone $\Delta\tau^k_{ij}(t)$ which depends on the quality of the solution found on the whole of its course. This algorithm has been adapted to our problem by replacing the city i by the part i and the city j by the routing j. For each part i, the choice of routing j is based on a compromise between the intensity of the trail $\tau^k_{ij}(t)$ and visibility η_{ij}

(this depends on the number of parts in the input buffer of the first machine of the routing and its load).

The relative importance of the two elements is always controlled by two coefficients *α* and *β*. If the full number of ants is *m* and the size of the loading station is *n*, a cycle is carried out when each of the *m* ants assigns *n* first parts of the infinite queue to routings *j*. After a full rotation (the assignment of all *n* first parts of the infinite queue to the routings by the ants), each ant leaves a certain quantity of pheromone $\Delta\tau^k_{ij}(t)$ which depends on the quality of the found solution on the whole of the selected routings for the parts. The algorithm is presented as follows:

```
Step 1: If there is a free place in the loading station then
Step 2: For $t = 1$ to $t_{\max}$
Step 3: For each ant $k = 1$ to $m$
Step 4: Select randomly a routing for the first part of the infinite queue according
        to its type
Step 5: For each part $i$ contents in the second place until the $n$th place of the infi-
        nite queue
Step 6: Select a routing $i$, among the possible routings according to a probability
        depending on the intensity of the trace and the number of the parts in the
        input queue of the first machine of this routing and its load
Step 7: End for
Step 8: Evaluation: of the objective function. (Produced loads of the routings)
Step 9: Leave a track $\Delta\tau^k_{ij}(t)$ on the way $T^k(t)$ (for each routing $j$ selected for part
        $i$ by the ant $k$)
Step 10: End for
Step 11: Evaporate the tracks and modify the intensities
Step 12: End for
Step 13: End if
```

8.5.2 Simulated Annealing

The SA method was developed by Kirkpatrick *et al.* (1983). It is a meta-heuristic inspired by a process used in metallurgy to obtain a well-ordered solid state with minimal energy called the annealing process. This technique consists in heating material to a high temperature, then decreasing this temperature slowly. This optimization method is based on work of Metropolis *et al.* (1953) which allows describing the behaviour of a system in thermodynamic equilibrium at a certain temperature. This technique transports the annealing process to the resolution of an optimization problem, the objective function to be minimized being the energy *E* of the material. The temperature *T* is also introduced. From an initial solution at a temperature *T*, we generate another close solution in a random way. If this solution improves the objective function, this latter is automatically accepted. If it degrades the objective function, it can also be accepted according to a probability $\exp(-\Delta E)$ where ΔE is the variation of the objective function, once thermodynamic equilibrium is reached, one should slightly reduce the front temperature before implementing a new iteration. The algorithm of this meta-heuristic is presented as follows:

```
Step 1:  If there is a free place in the loading station then
Step 2:  Build the initial state (assign $n$ first parts to routings randomly)
Step 3:  Calculate the product of loads of the routings
Step 4:  For $t = 1$ to $t_{\max}$
Step 5:  Modify the routings of certain parts among $n$ first parts contained in the
         infinite queue
Step 6:  Calculate the product of loads of the routings
Step 7:  If the objective function is improved then this solution is accepted
Step 8:  End if
Step 9:  Else, generate a random number
Step 10: If this number is lower or equal to $\exp(-\Delta E)$: ($\Delta E$ is the variation of the
         objective function) then this solution is accepted
Step 11: End if
Step 12: End if
Step 13: End for
Step 14: End
```

8.5.3 Particle Swarm Optimization

PSO is a recent meta-heuristic approach proposed by Eberhart and Kennedy (1995). It is based on the metaphor of social interaction and communication, such as fish schooling and bird flocking when they are randomly searching for food in an area, where there is only one piece of food available and none of them knows where it is, but they can estimate how far it would be at each iteration. For this problem, the simplest strategy to find and get the food is to follow the bird known to be the nearest one to the food.

In PSO, each single solution is called a particle, the group becomes a swarm (population) and the search space is the area to explore. Each particle has a fitness value calculated by a fitness function, and a velocity of flying towards the optimum. In the original version of PSO, all particles adjust their positions not only according to their own experience but also according to the experience of other particles; they fly across the problem space following the particle nearest to the optimum by two elastic forces. One attracts it to the best location so far encountered by the particle. The other attracts it with random magnitude to the best location encountered by any member of the swarm. In the rest of this section we use this notation:

$X_i(t)$:	the position of particle i at t iterations;
$\lambda_i(t)$:	the velocity of the particle;
$\delta_i(t)$:	cognition part of the particle;
n_1:	number of individuals;

p_i:	the best local;
G:	the best global;
F_1:	operator of modification;
F_2, F_3:	operators of crossing;
C_1:	cognitive coefficient;
C_2:	social coefficient;
W:	probability of modification.

PSO was originally designed for continuous optimization problems, but can be adapted to discrete problems like our problem of routing selection where the position of the particle is updated by the following equation proposed by Pan *et al.* (2005):

$$X_i(t) = c_2 \oplus F_3\left(c_1 \oplus F_2\left(w \oplus F_1\left(X_i(t-1)\right), p_i(t-1)\right), G(t-1)\right). \tag{8.6}$$

Equation 8.6 consists of three components:

- The first component is $\lambda_i(t) = w \oplus F_1(X_i(t-1))$ which represents the velocity of the particle. F_1 represents an operator, which modifies the routing of some parts with the probability of w, a uniform random number r is generated between 0 and 1. If r is less than w then the F_1 is applied to generate a perturbed permutation of the particle by $\lambda_i(t) = F_1(X_i(t-1))$, otherwise current permutation is kept as $\lambda_i(t) = X_i(t-1)$.
- The second component is $\delta_i(t) = c_1 \oplus F_2(\gamma_i(t), p_i(t-1))$, it represents the cognition part of the particle.
- The third component is $X_i(t) = c_2 \oplus F_3(\delta_i(t), G_i(t-1))$, it represents the social part of the particle.

F_2 and F_3 represent the crossover with the probability C_1 and C_2, and p_i and G are the best local and global. The algorithm of particle swarm optimization adapted for routing selection is as follows:

```
Step 1: $n$ is the size of the queues
Step 2: If there is a free place in the loading station then
Step 3: Initialize the population
Step 4: Initialize the parameters
Step 5: While the algorithm has not met stop criterion
Step 6: For $i = 1$ to $n_1$ (for each particle $x_i$)
Step 7: Calculate the products of the routing loads of this particle
Step 8: If $F(x_i) > F(p_i)$ then: update the best local $p_i = x_i$
Step 9: End if
Step 10: If $F(x_i) > F(G)$ then: update the best global $G = x_i$
Step 11: End if
Step 12: End for
Step 13: For $i = 1$ to $n_1$ (update the particle position)
```

```
                $X_i(t) = c_2 \oplus F_3(c_1 \oplus F_2(w \oplus F_1(X_i(t-1)), p_i(t-1)), G(t-1))$
Step 14: End for
Step 15: End while
Step 16: End if
Step 17: End
```

8.5.4 Genetic Algorithms

GAs were proposed by Holland (1975). They were inspired from the principles of natural genetics and the theory of evolution. In a GA, each solution is stored in an artificial chromosome represented by a code. Each of these chromosomes is defined by two characteristics. The first is their genotype, which is the actual sequence, which defines the chromosome. It is so called because of the analogy with a genetic sequence in biology. The second is the phenotype, which is the decoded version of the genotype that determines the traits of the individual.

With each of the chromosomes, the parameters are decoded and evaluated by the fitness function to determine the quality of the phenotype. New candidates are generated gradually from a set of renewed populations by applying artificial genetic operators selected after, repeatedly using operators of crossover and mutation (Goldberg, 1989). Crossover is performed by taking two bit genotypes, choosing a place along the bit string, cutting each of them at that place and then connecting one string's left to the other string's right and *vice versa*. This produces two new chromosomes, which are a combination of the two parents.

Reproduction is simply a matter of selecting chromosomes which are judged to be above a certain fitness level for the next generation and mutation is done by choosing bits randomly and swapping them. The adaptation of GAs is given as follows:

```
Step 1:  If there is a free place in the loading station then
Step 2:  Generate a random population
Step 3:  While the algorithm has not met stop criterion
Step 4:  For each individual
Step 5:  Evaluate the fitness of this individual (the product of the routing loads)
Step 6:  If the objective function is higher than the best solution then update the
         best solution
Step 7:  End for
Step 8:  Select the individuals for the reproduction (selection operator)
Step 9:  Apply the operator of crossing (we obtain a set of new individuals)
Step 10: Apply the operator of mutation on the new individuals
Step 11: Constitute the new generation
Step 12: End while
Step 13: End if
```

8.5.5 Taboo Search

This method was formalized by Glover and Laguana (1997). It is based on the use of mechanisms inspired by the human memory. An algorithm based on this meta-heuristic requires an initial solution and a neighbourhood structure.

The principle of this meta-heuristic is simple: we generate an initial configuration and we proceed by transiting from one solution to another. The mechanism of transition from one configuration, called s, to the next one, called t, consists of two stages (Dréo *et al.*, 2003):

- The first builds the set of the neighbours of s, i.e. the set of the accessible configurations in only one elementary movement of s. Let $V(s)$ be the set (or the subset) of these neighbours.
- The second evaluates the objective function f of the problem for each configuration belonging to $V(s)$. The configuration t, which succeeds s in the series of the solutions built by the taboo method, is the configuration of $V(s)$ in which f takes the minimal value. This configuration t is adopted even if it is worse than s; due to this characteristic the taboo method can avoid the trapping in the local minima.

To avoid returning to a retained configuration and generating a loop, the taboo list is created. This taboo list, of fixed or variable length, which gave its name to the method, contains m movements ($t \to s$), which are the opposite of the last m movements ($s \to t$) carried out. The movements stored in the taboo list are forbidden. The algorithm of this meta-heuristic is presented as follows:

```
Step 1:  If $n$ is the capacity of the queues
Step 2:  If there is a free place in the loading station then
Step 3:  Build the initial state $s$ (assign $n$ first parts of the infinite queue to rout-
         ings in a random way)
Step 4:  Initialize the parameters
Step 5:  Calculate the products of the routings loads
Step 6:  While the algorithm has not met stop criterion
Step 7:  For $t$ = 1 to $n$_neighbours
Step 8:  Modify the routings of certain parts chosen randomly among $n$ first parts
         of the infinite queue with no taboo movements (modification of $s$)
Step 9:  Calculate the products of the routings loads (objective function)
Step 10: If the objective function is higher than the best solution, then update the
         best solution
Step 11: End if
Step 12: End for
Step 13: If $t$ is the best of these neighbours then
Step 14: Insert movement $t \to s$ in the taboo list
Step 15: $t = s$
Step 16: End while
Step 17: End if
```

8.5.6 Electromagnetism-like Method

Proposed by Birbil and Fang (2003), the EM is a population-based meta-heuristic dedicated to solving effectively continuous problems. The EM simulates the attraction–repulsion mechanism of electromagnetism theory, which is based on Coulomb's law. For this approach, each solution is characterized and updated by a charge and a force, the charge of each particle is relative to the objective function. The generic pseudo-code for the EM, which consists of five procedures, is presented as follows:

```
1. Initialize population.
2. While the algorithm has not met stop criterion.
3. Local search.
4. Calculate total force $F$.
5. Move particle by $F$.
6. Evaluate particles.
7. End While.
```

To solve our problem we used a hybrid framework, which combines the EM algorithm with genetic operators proposed by (Chen *et al.*, 2007). The following part presents the algorithm of Electromagnetism-like method meta-heuristics adapted for routing selection.

```
Step 1:  If there is a free place in the loading station then
Step 2:  Generate a random population
Step 3:  While the algorithm has not met stop criterion
Step 4:  For $i = 1$ to $m$ ($m$: size of the population)
Step 5:  Search with modifying the routings of parts among the $n$ first parts of the
         infinite queue
Step 6:  Evaluate objective function
Step 7:  If the objective function is higher than the best solution then update the
         best solution
Step 8:  End for
Step 9:  Calculate the objective functions average $avg$
Step 10: For $i = 1$ to $m$
Step 11: If $I \neq$ best and $F(X_i) > avg$ then
Step 12: $j$ = particle selected
Step 13: Uniform crossing ($X_i$, $X_j$)
Step 14: End if
Step 15: Else If $F(X_i) < avg$ Then
Step 16: Calculate force and move ($X_i$)
Step 17: End if
Step 18: End for
Step 19: End while
Step 20: End if
```

8.6 Performance Evaluation of Routing Selection Methods

The main objective of this section is the comparative study of performance, for routing selection, between the meta-heuristics and modified DMM presented before. The study is realized in two parts: with and without presence of breakdown in the system. The studied performance evaluation criteria are: production rate, machine utilization rate, material handling utilization rate, work-in-process and cycle time.

8.6.1 System Simulation Without Presence of Breakdown

8.6.1.1 Production Rate

Table 8.2 and Figure 8.3 state that for a significant rate of arrival of parts in the system (between 5 and 20 minutes), production rates obtained by meta-heuristics are practically the same of those obtained with modified DMM with a little advantage for meta-heuristics. If the arrival rate is less than or equal to 1/25 the production rate is practically the same for all methods. We can see that the results of PSO and GA meta-heuristics are the best among meta-heuristics, particularly if the rate of parts arrival is greater than or equal to 1/20.

Table 8.2 Production rate for queue size = 2

Rate of arrival of the parts (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Modified DMM	99.99	99.99	99.98	99.71	84.47	60.73	41.67	21.15
ACO	99.99	99.99	99.99	99.99	90.52	64.54	43.20	21.51
TS	99.99	99.99	99.99	99.99	95.28	64.93	43.29	21.78
GA	99.99	99.99	99.99	99.99	98.44	71.08	47.25	23.70
PSO	99.99	99.99	99.99	99.99	99.22	69.33	46.10	23.12
SA	99.99	99.99	99.99	99.99	94.73	61.90	41.21	20.61
EM	99.99	99.99	99.99	99.99	98.52	66.94	44.75	22.42

8.6.1.2 Machine Utilization Rate

The utilization rate of machines is a very important criterion in performance measurement of production systems. The utilization rates for the machines VTC1 and VTC2 are almost the same with a small advantage to the GA meta-heuristic for high part arrival rate and to modified DMM for low arrival rate (Table 8.3).

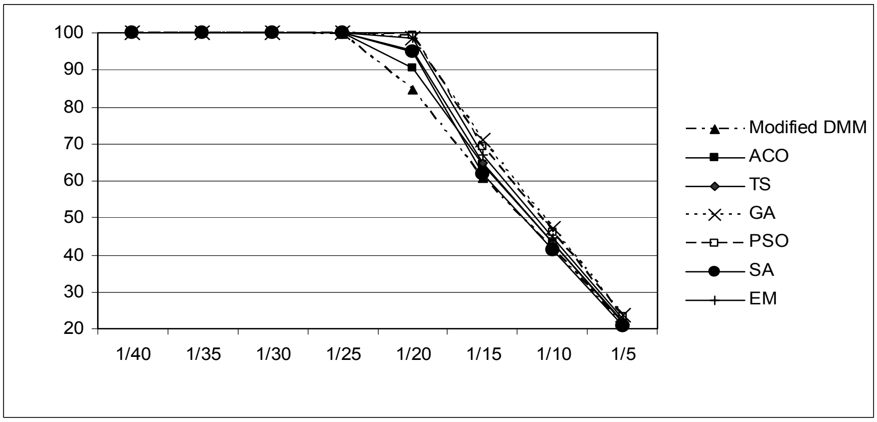


Fig. 8.3 Production rate for queue size = 2

Table 8.3 VTC machine utilization rate for queue size = 2

Rate of arrival of the parts (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Modified DMM	49.97	56.94	66.59	79.85	84.52	82.72	83.58	84.70
ACO	48.44	54.59	63.99	76.31	85.73	81.97	82.26	82.06
TS	48.24	54.42	63.72	76.43	90.36	84.73	84.75	85.27
GA	48.36	54.51	64.32	75.67	92.72	90.98	90.85	90.96
PSO	48.29	54.71	63.96	76.55	93.20	89.20	88.84	88.95
SA	48.59	54.74	64.16	76.2	89.42	80.49	80.41	80.47
EM	48.24	54.48	63.96	76.19	92.39	86.16	86.56	86.75

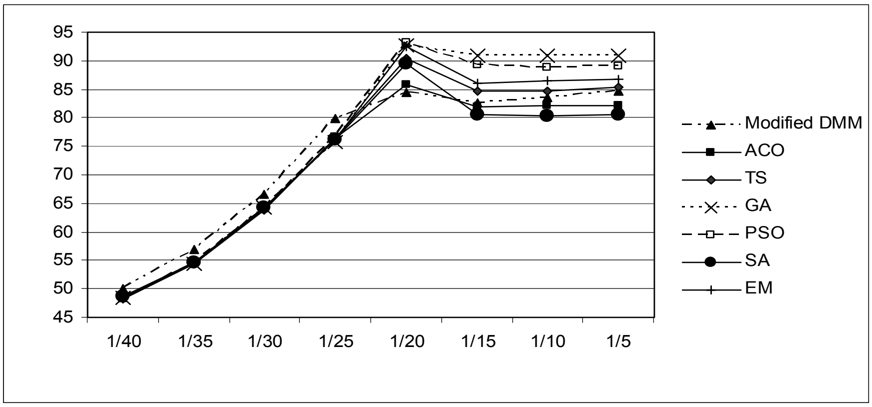


Fig. 8.4 VTC machine utilization rate for queue size = 2

8.6.1.3 Material Handling Utilization Rate

Table 8.4 and Figure 8.5 show us that for a saturated system (rate of arrival of parts over 1/25) the utilization rate of the AGV is practically the same for all methods. The results given by meta-heuristics are less good than those of the modified DMM but these results are more stable because variations of values are very low; this is due to the high production rate and the increase in the use of machines. The results of PSO and GA meta-heuristics are little better than other meta-heuristics if the system is saturated.

Table 8.4 Material handling utilization rate for queue size = 2

Rate of arrival of the parts (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Modified DMM	14.98	17.46	21.00	27.31	30.26	29.16	30.19	30.44
ACO	17.47	19.58	22.89	27.36	30.64	29.37	29.46	29.41
TS	17.37	19.50	22.86	27.42	32.32	30.68	30.69	30.88
GA	17.43	19.54	23.16	27.04	33.07	32.69	32.67	32.68
PSO	17.40	19.64	22.98	27.48	33.20	32.00	31.97	31.98
SA	17.55	19.65	23.08	27.30	31.92	29.10	29.07	29.10
EM	17.37	19.52	22.98	27.29	32.89	31.12	31.20	31.26

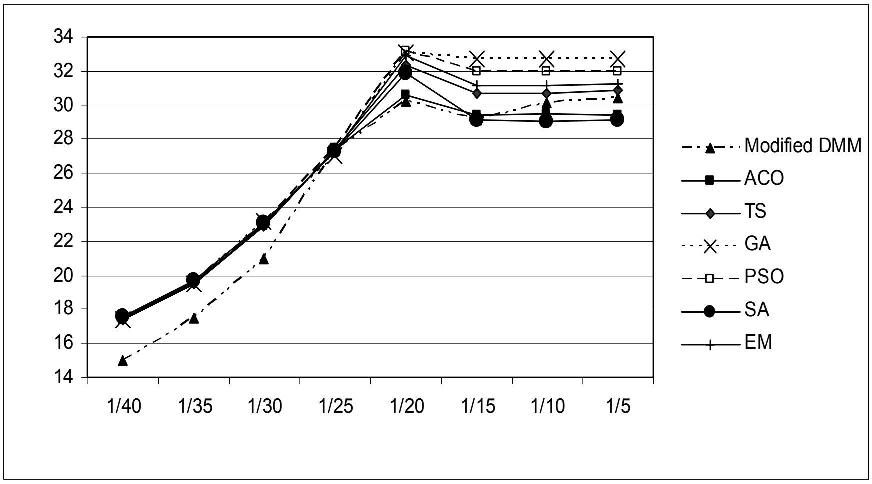


Fig. 8.5 Material handling utilization rate for queue size = 2

8.6.1.4 Work-in- Process

Table 8.5 and Figure 8.6 show that the number of parts that remain in the system if we use modified DMM is higher than those if we use the meta-heuristics. If the rate of parts arrival is greater than 1/25 then the GAs give the best results among the other meta-heuristics. If the rate of parts arrival is smaller than 1/20, then the meta-heuristics give practically the same results, however, the results given by modified DMM are much better than those of meta-heuristics.

Table 8.5 Work-in-process for queue size = 2

Rate of arrival of the parts (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Modified DMM	6.79	12.9	15.8	19.0	17.4	18.2	16.9	18.1
ACO	3.90	4.04	4.45	5.21	7.29	8.40	8.44	8.46
TS	3.89	4.03	4.47	5.26	7.98	8.40	8.40	8.41
GA	3.90	4.04	4.46	5.23	7.45	8.90	8.85	8.85
PSO	3.88	4.03	4.45	5.25	7.35	8.67	8.67	8.64
SA	3.91	4.03	4.46	5.22	7.11	7.95	7.95	7.95
EM	3.89	4.04	4.44	5.28	7.99	8.64	8.66	8.62

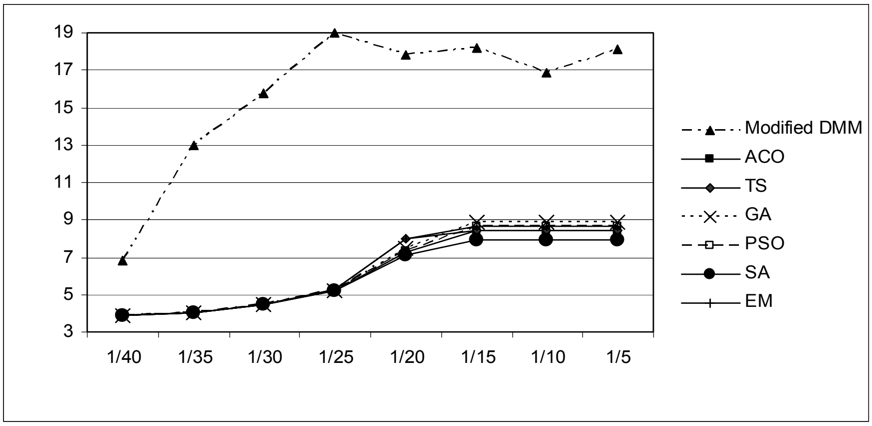


Fig. 8.6 Work-in-process for queue size = 2

8.6.1.5 Cycle Time

Table 8.6 and Figure 8.7 show that if the rate of arrival of the parts is greater than 1/30, the cycle times for meta-heuristics cases are smaller than the ones for modified DMM. The results of cycle time given by meta-heuristics are practically the same. If the rate of parts arrival is between 1/30 and 1/40 we can see that the results of all methods are similar. These results show that the cycle times given by meta-heuristics are better than those of modified DMM in most cases.

Table 8.6 Cycle time for queue size = 2

Rate of arrival of the parts (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Modified DMM	81.94	87.83	101.63	155.9	203.4	204.7	207.2	204.2
ACO	90.91	83.76	89.03	98.83	155.3	166.7	165.4	163.6
TS	91.49	80.37	86.24	98.02	136.0	171.8	174.8	173.9
GA	89.62	81.81	88.65	98.74	132.7	169.9	168.9	168.1
PSO	89.62	81.23	91.70	96.19	125.9	173.5	173.2	169.5
SA	95.33	83.81	90.34	99.59	135.5	170.4	168.9	168.2
EM	92.29	81.6	90.16	101.1	124.8	170.6	170.9	168.9

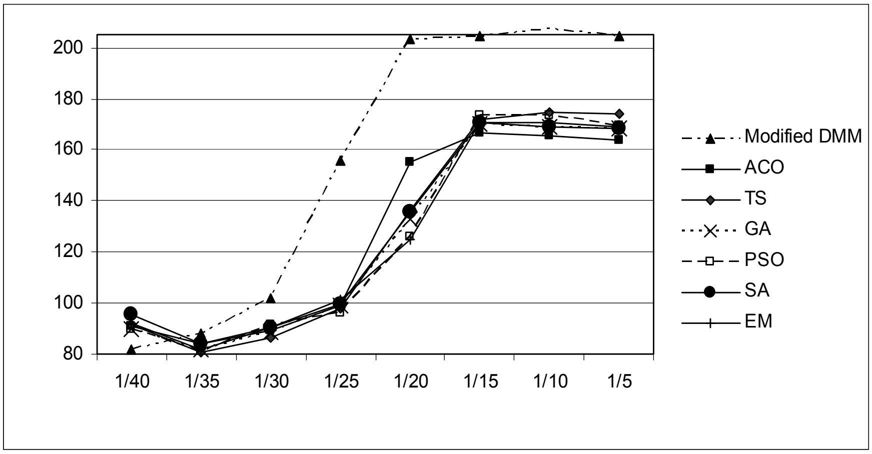


Fig. 8.7 Cycle time for queue size = 2

8.6.2 System Simulation With Presence of Breakdown

8.6.2.1 Production Rate

The results of Table 8.7 and Figure 8.8 are similar to those in Table 8.2 and Figure 8.3. For a rate of arrival of parts in the system between 1/5 and 1/25, production rates obtained by meta-heuristics are better than those obtained by modified DMM. GA and PSO meta-heuristics give the best results. If the arrival rate is less or equal then 1/30 the production rate is practically the same for all methods. We can conclude that in general even with the presence of breakdown in the system, the meta-heuristics perform the best.

Table 8.7 Production rate for queue size = 2

Rate of arrival of the parts (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Modified DMM	99.99	99.97	99.62	85.02	66.70	50.91	34.29	17.04
ACO	99.99	99.99	99.99	99.98	84.47	63.13	42.11	21.13
TS	99.99	99.99	99.99	99.99	84.53	63.45	42.28	21.12
GA	99.99	99.99	99.99	99.99	92.00	69.05	46.22	23.12
PSO	99.99	99.99	99.99	99.99	90.04	67.50	44.99	22.61
SA	99.99	99.99	99.99	99.99	80.72	60.22	40.24	20.23
EM	99.99	99.99	99.99	99.99	89.78	65.61	43.76	21.96

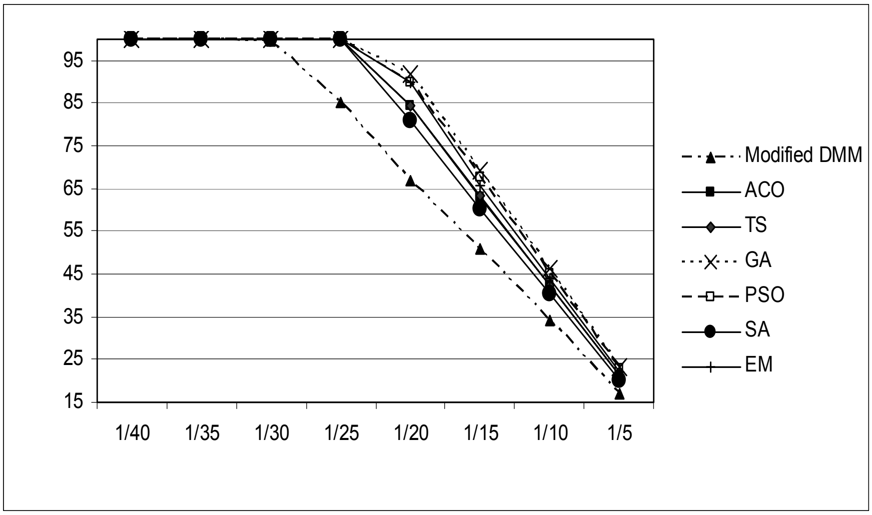


Fig. 8.8 Production rate for queue size = 2

8.6.2.2 Machine Utilization Rate

Even with the introduction of failures the utilization rates for the machines VTC1 and VTC2 are more significant for meta-heuristics when compared to modified DMM if the system is saturated (rate of arrival of parts is over than 1/20). Otherwise, the rates of utilization of the VTC machines are almost similar for all methods. Table 8.8 and Figure 8.9 show that PSO and GA meta-heuristics are the best if the system is saturated.

Table 8.8 VTC machine utilization rate for queue size = 2

Rate of arrival of the parts (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Modified DMM	49.88	56.94	66.78	77.16	77.77	78.13	78.50	78.75
ACO	48.29	54.58	63.90	76.13	80.44	80.20	80.24	80.50
TS	48.40	54.65	63.96	76.03	82.75	82.80	82.76	82.75
GA	48.30	54.90	64.04	76.10	88.13	88.44	88.63	88.67
PSO	48.64	54.80	64.26	76.45	86.71	86.63	86.61	86.90
SA	48.23	54.47	64.25	76.20	78.73	78.35	78.53	79.84
EM	48.30	54.60	64.01	75.95	86.16	84.60	84.65	84.96

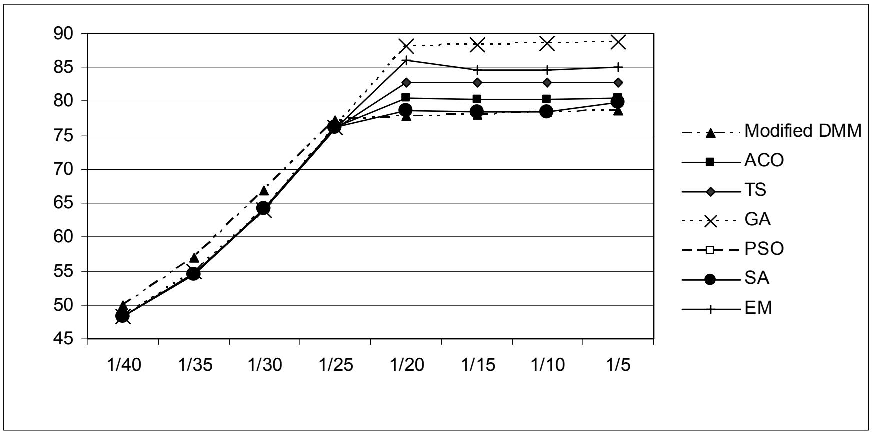


Fig. 8.9 VTC machine utilization rate for queue size = 2

8.6.2.3 Material Handling Utilization Rate

Table 8.9 and Figure 8.10 demonstrate that for a saturated system the utilization rate of the AGV is more significant for meta-heuristics than the modified DMM rule. If the rate of arrival of parts is less than 1/25 all of the methods give practically the same performance. The results of PSO and GA meta-heuristics are little better than other meta-heuristics if the system is saturated.

8.6.2.4 Work-in-Process

Table 8.10 and Figure 8.11 demonstrate that if the rate of arrival of parts is greater or equal than 1/30, the number of parts that stay in the system if we use modified DMM is higher than those of meta-heuristics. In this case the results of all meta-heuristics are practically the same. If the system is not saturated the results given by meta-heuristics and modified DMM are very similar.

Table 8.9 Material handling utilization rate for queue size = 2

Rate of arrival of the parts (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Modified DMM	14.98	17.55	21.19	24.28	24.07	24.54	24.67	24.54
ACO	17.39	19.57	22.95	27.27	28.82	28.74	28.75	28.84
TS	17.45	19.61	22.97	27.21	29.96	29.98	29.96	29.97
GA	17.40	19.73	23.02	27.25	31.78	31.79	31.83	31.85
PSO	17.57	19.68	23.13	27.42	31.20	31.16	31.15	31.24
SA	17.37	19.52	23.13	27.30	28.46	28.33	28.31	28.88
EM	17.41	19.58	23.01	27.17	30.96	30.49	30.51	30.62

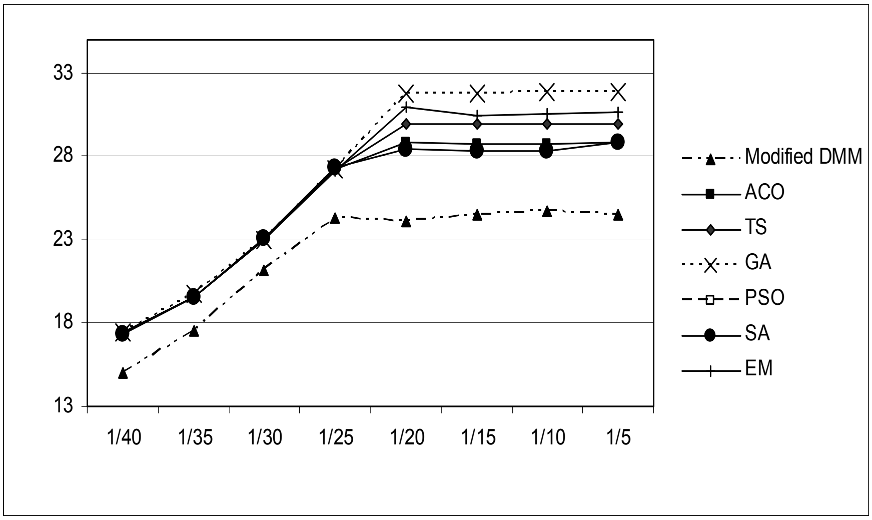


Fig. 8.10 Material handling utilization rate for queue size = 2

Table 8.10 Work-in-process for queue size = 2

Rate of arrival of the parts (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Modified DMM	4.55	7.79	28.6	29.8	30.5	28.0	28.6	27.2
ACO	3.95	4.09	4.55	5.40	8.46	8.43	8.43	8.46
TS	3.94	4.10	4.54	5.36	8.41	8.40	8.41	8.39
GA	3.92	4.10	4.58	5.34	8.83	8.87	8.85	8.81
PSO	3.94	4.09	4.55	5.39	8.67	8.68	8.68	8.70
SA	3.93	4.10	4.54	5.44	7.95	7.94	7.99	8.05
EM	3.93	4.09	4.56	5.41	8.65	8.61	8.61	8.60

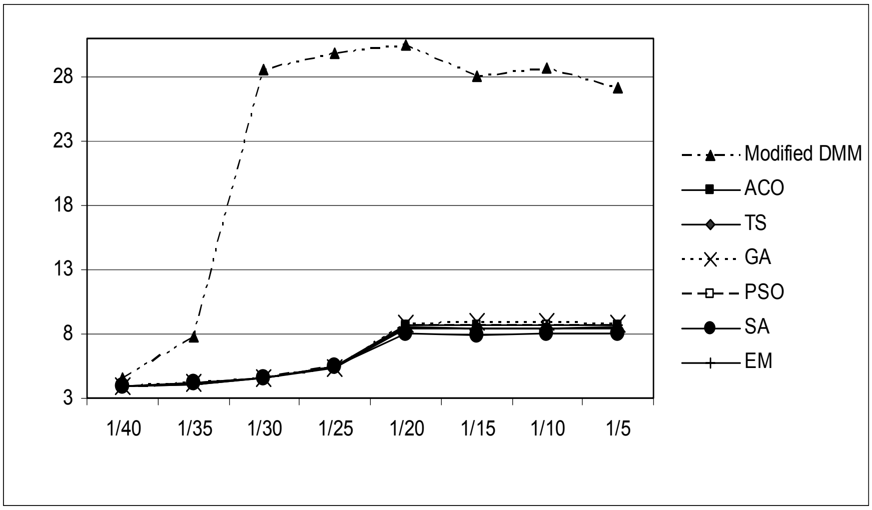


Fig. 8.11 Work-in-process for queue size = 2

8.6.2.5 Cycle Time

Table 8.11 and Figure 8.12 show that if the rate of arrival of parts is greater than 1/30, the cycle time of these parts in the system is smaller in the meta-heuristics case than in the modified DMM case. The results of cycle time given by meta-heuristics are practically the same. If the rate of arrival of the parts is between 1/30 and 1/40 we can see that the results of all methods are similar. These results show that the cycle times given by meta-heuristics are better than those of modified DMM in the case of a saturated system even with presence of breakdowns.

Table 8.11 Cycle time for queue size = 2

Rate of arrival of the parts (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Modified DMM	91.10	99.13	120.0	325.2	353.3	346.8	339.9	341.1
ACO	97.59	82.67	90.14	104.0	169.9	169.9	167.4	175.4
TS	94.59	85.92	91.91	103.4	177.1	178.4	173.8	176.1
GA	94.27	85.76	93.30	104.6	170.8	174.6	175.2	176.3
PSO	95.50	84.90	93.20	103.2	170.8	175.1	174.6	173.9
SA	93.07	86.26	93.58	99.45	172.3	175.8	175.1	173.4
EM	92.78	83.37	91.26	102.7	174.1	172.9	175.8	169.9

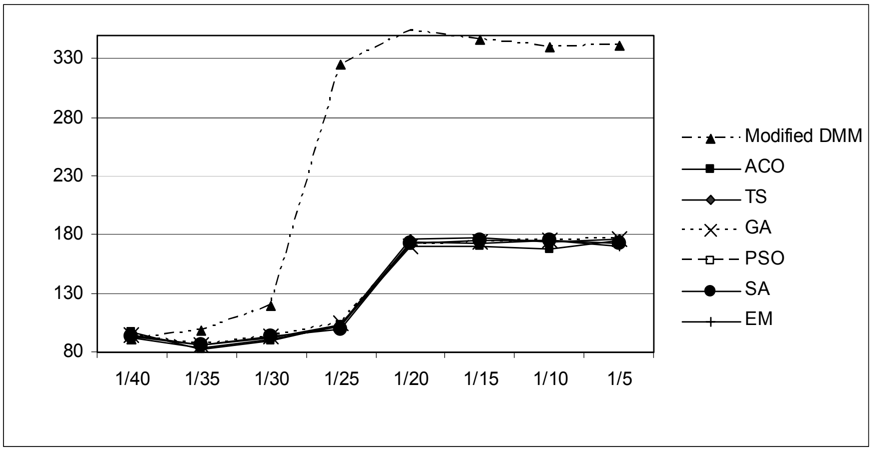


Fig. 8.12 Cycle time for queue size = 2

8.7 Conclusions

In this chapter we presented a problem of job-shop routing scheduling. For that, six meta-heuristics: ACO, SA, PSO, GAs, TS and EM were compared to the modified DMM which is a rule of selection of alternative routing in real time. This comparison study was performed on a job-shop model picked from the literature. The system was simulated with and without breakdown. Five criteria were selected for performance evaluation and comparison, namely: production rate, machine utilization rate, material handling utilization rate, work-in-process, and cycle time.

For each rule, we notice that the simulation results without breakdowns are better than those with breakdown, which is predictable since breakdown lowers the performance of the system. Results obtained showed that all meta-heuristics produced better results than modified DMM. Also, they clearly increased the performance of the system for a saturated production system. This is because they increase the production rate and the utilization rate of machines and the utilization of AGV. We can note the improvement of the performance concerning the cycle time and the rate of the work-in-process of the system if we use the meta-heuristics when compared to modified DMM.

Results showed that PSO and GA generate the best results practically in all cases. However, if the production system is not overloaded, all methods produce practically the same results. The performance and results were obtained after many replications and simulation with ARENA and Java software for a duration of 20,000 hours for each replication.

From this work, we can conclude that meta-heuristics are suitable for real-time selection of routing in a job-shop. The computation time for one routing selection varies from one method to another but in all cases this time is of the order of milliseconds. So the computation time does not influence the selection of one method among others.

References

Basnet C, Mize JH (1994) Scheduling and control of flexible manufacturing systems: a critical review. Int. Comput. Integr. Manuf., 7(6):340–355

Birbil SI, Fang S (2003) An electromagnetism like mechanism for global optimization. J. Glob. Opt., 25:263–282

Blum C, Roli A (2003) Meta-heuristics in combinatorial optimization: overview and conceptual Comparison. ACM Comput. Survey, 35:268–308

Byrne MD, Chutima P (1997) Real-time operational control of an FMS with full routing flexibility. Int. J. Prod. Econ., 51:109–113

Chen S, Chang P, Chan C et al. (2007) A hybrid electromagnetism-like algorithm for single machine scheduling problem. In D.-S. Huang et al. (eds) Advanced intelligent computing theories and applications. with aspects of artificial intelligence. Springer

Cho H, Wysk RA (1995) Intelligent workstation controller for computer-integrated manufacturing: problems and models. J. Manuf. Syst., 14(4):252–263

Das SK, Nagendra P (1997) Selection of routes in a flexible manufacturing facility. Int. J. Prod. Econ., 48:237–247

Dorigo M (1992) Optimization, learning and natural algorithms, PhD thesis, DEI, Politecnico di Milano, Italy

Dréo J, Pétrowski A, Siarry P et al. (2003) Métaheuristiques pour l'optimisation difficile. Eyrolles

Eberhart RC, Kennedy J (1995) New optimizer using particle swarm theory. In: Proceedings of the 6th International Symposium on Micro Machine and Human Science, pp. 39–43

Garey MR, Johson DS (1979) Computers and intractability a guide of the theory of NP- completeness. W.H. Freeman and Company, San Francisco, USA

Glover F, Greenberg HJ (1989) New approaches for heuristic search: A bilateral linkage with artificial intelligence. Eur. J. Oper. Res., 39:119–130

Glover F, Laguana M (1997) Tabu search. article principally adapted. In Glover and Laguana (eds) Tabu search. Kluwer Academic

Goldberg EE (1989) Genetic algorithms in search, optimization, and machine learning. Addison Wesley, Reading, MA

Gupta Y P, Gupta MC, Bector CR (1989) A review of scheduling rules in flexible manufacturing systems. Int. J. Comp. Integr. Manuf., 2:356–377

Hassam A, Sari Z (2007) Real-time selection of process plan in flexible manufacturing systems: Simulation study. In: Proceedings of the International Conference on Industrial Engineering and Systems Management, Beijing, China

Holland JH (1975) Adaptation in natural and artificial systems. University of Michigan Press

Ishii N, Muraki M (1996) A process-variability-based on-line scheduling system in multi product batch process. Comput. Chemi. Eng., 20:217–234

Kazerooni A, Chan FTS, Abhary K (1997) A fuzzy integrated decision-making support system for scheduling of FMS using simulation. Comput. Integr. Manuf. Syst., 10(1):27-34

Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science, 220(4598):671–680

Kouiss K, Pierreval H, Mebarki N (1997) Using multi-agent architecture in FMS for dynamic scheduling. J. Intell. Manuf., 8:41–47

Liu J, MacCarthy BL (1997) A goal MILP model for FMS scheduling. Eur. J. Oper. Res., 100: 441–453

Metropolis N, Rosenbluth AW, Rosenbluth MN et al. (1953) Equations of state calculations by fast computing machines. J. Chem. Phys., 21:1087–1091

Nof S, Barash M, Solberg J (1979) Operational control of item flow in versatile manufacturing system. Int. J. Prod. Res., 17:479–489

Pan QK, Tasgetiren MF, Liang YC (2005) A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem with makespan criterion. In: Proceedings of the International Workshop on Planning and Scheduling Special Interest group, UK PLANSIG2005. City University, London, pp. 31–41

Peng C, Chen FF (1998) Real-time control and scheduling of flexible manufacturing systems: a simulation based ordinal optimization approach. Int. J. Adv. Manuf. Tech., 14(10):775–786

Rachamadugu R, Stecke KE (1994) Classification and review of FMS scheduling procedures. Prod. Plan. Control, 5:2–20

Saygin C, Kilic SE (2004) Dissimilarity maximization method for real-time routing of parts in random flexible manufacturing systems. Int. J. Flex. Manuf., Syst., 16(2):169–182

Saygin C, Chen FF, Singh J (2001) Real-time manipulation of alternative routings in flexible manufacturing systems: A simulation study. Int. J. of Adv. Manuf. Tech., 18:755–763

Saygin C, Kilic SE (1996) Effect of flexible process plans on performance of flexible manufacturing systems. In: Proceedings of 7th International DAAM symposium, Vienna, Austria, pp.. 393–394

Saygin C, Kilic SE (1997) Scheduling of flexible manufacturing system. In: Proceedings of MicroCAD 97 Conference, University of Miskolc, Hungary, vol. H, pp. 19–23

Saygin C, Kilic SE (1999) Integrating flexible manufacturing systems with scheduling in flexible manufacturing system. Int. J. Adv. Manuf. Tech., 15(4):268–280

Saygin C, Kilic SE, Toth T et al. (1995) On scheduling approaches of flexible manufacturing systems: gap between theory and practice. In: Proceedings of the 3rd IFAC/IFIP/IFORS Workshop – Intelligent Manufacturing Systems 95, Pergamon, pp. 61–66

Shukla C S, Chen FF (1996) The state of the art in intelligent real-time FMS control: a comprehensive survey. J. Intell. Manuf., 7:441–455

Wu SYD, Wysk RA (1989) An application of discrete event simulation to on-line control and scheduling in flexible manufacturing. Int. J. Prod. Res., 27:1603–1623

Chapter 9
Meta-heuristic Approaches for Multi-objective Simulation-based Optimization in Supply Chain Inventory Management

D. Sánchez[1], L. Amodeo and C. Prins[2]

Abstract A supply chain (SC) is a complex network of facilities with dissimilar and conflicting objectives. Discrete-event simulation is often used to model and capture the dynamic interactions in SCs and to provide performance indicators. However, a simulator by itself is not an optimizer. Optimization algorithms can be coupled with a simulation module in order to find the most suitable SC policies. Nevertheless, because the simulation of an SC can take a considerable amount of time, the optimization tool must be well chosen. This chapter considers the hybridization of evolutionary algorithms, well known for their multi-objective capabilities, with an SC simulation module in order to determine the inventory policy (order-point or order-level) of a single product SC, taking into account two conflicting objectives: maximizing customer service level and minimizing total inventory cost. Four algorithms (SPEA-II, SPEA-IIb, MOPSO and NSGA-II) are evaluated on five different SC configurations to determine which algorithm gives the best results and makes the best use of the simulator. The results indicate that SPEA-2 favours a rapid convergence and that modifying its crossover or its archive truncation rule (variant SPEA-IIb) may improve the results even further.

Keywords Supply chain management, simulation, multi-objective optimization, meta-heuristics, evolutionary algorithms

9.1 Introduction

Nowadays, it is impossible for one manufacturer alone to completely produce complex products. Therefore, in a supply chain (SC), separate manufacturers are

[1] D. Sánchez
Centro de Optimización et Probabilidad Aplicada (COPA), Departamento de Ingeniería Industrial, Universidad de los Andes, Carrera 1 N° 18A-10 A.A. 4976, Bogotá, DC, Colombia

[2] L. Amodeo and C. Prins (✉)
ICD-LOSI, Université de Technologie de Troyes, BP 2060, 10010 Troyes Cedex, France
e-mail: christian.prins@utt.fr

in charge of a subset of the chain that must be managed and run efficiently. The efforts to coordinate the members to act for the common good have led to the science known as supply chain management (SCM). SCM is important as its main goal is to improve operational issues such as manufacturing and procurement processes. A significant amount of research deals with the operational issue of *inventory control*. Since the holding of inventories in an SC can cost between 20% and 40% of the final product value, deciding the *level of inventories* to be maintained and the *order-up-to level* in different stages of the SC is critical for minimizing the total cost of the system and maximizing the service level to customers.

Another challenge when analyzing an SC is the modelling of interactions between echelons. As it is impossible to handle all dynamically changing SC variables using analytical methods, *discrete-event simulation* is well suited for dealing with stochastic variables since it can model multivariate and non-linear relations. Nonetheless, in most cases, the study of the performance of the SC goes in hand with the need to optimize network performance. Alone, a simulation model is not an optimizer but, by introducing a *search procedure* coupled with the simulator, both aforementioned aims are achieved. This approach is known as *simulation-based optimization* (Fu, 2001): the output of the simulation is iteratively used by the optimization module to get feedback on the search for the optimum.

Several search techniques can be used in conjunction with a simulation module but *evolutionary algorithms* (EAs) are the most appealing. Indeed, some EAs like the *strength pareto evolutionary algorithm* (SPEA-II) and the *non-dominated sorting genetic algorithm* (NSGA-II) have a solid reputation in handling conflicting objectives of multi-objective optimization problems. Working on one population of solutions and in one single run, they are able to build quickly a near-optimal Pareto frontier. It is also interesting to test a more recent meta-heuristic like particle swarm optimization (PSO) because, to the best of our knowledge, it has never been used for simulation-based optimization. Although PSO is not considered as a genuine EA by some authors, it is clearly a population-based method.

The purpose of this chapter is to test the coupling of four EAs with an SC simulation module for finding the inventory policy in an SC with two conflicting objectives: the total inventory cost and the customer service level. Apart from solution quality, we wish to determine the algorithms which make the best use of the simulator. Indeed, since each call to the simulation module is time-consuming, the most adequate EA has to gain as much information as possible from every call. This evaluation is important when an effective and reasonably fast algorithm is required, not only for inventory policy determination but also for any application where simulation parameters are to be optimized.

The rest of the chapter is organized as follows. Sect. 9.2 cites relevant works and their contributions. The problem to be solved is detailed in Sect. 9.3. Sect. 9.4 shows how SPEA-II, a new variant called SPEA-IIb, a multi-objective PSO (MOPSO) and NSGA-II are implemented for our problem. Sect. 9.5 presents the evaluation criteria for algorithm selection, the parameters used and the main results. Finally, some concluding remarks are presented in Sect. 9.6.

9.2 Literature Review

An SC is a network of suppliers, manufacturers, distribution centres (or warehouses) and retailers, through which products or services are produced and delivered to customers. One important issue in SCM is inventory control. Most companies wish to reduce their inventory costs and improve their service level to customers, by optimizing the inventory policies of their SCs. This optimization is difficult in practice because SCs are multi-echelon inventory systems with multiple interrelated stocks and conflicting objectives. For example, the stocks of raw materials or components of a manufacturer depend on the stocks of its suppliers, and reducing inventory cost and improving service level are two conflicting objectives.

In the literature, SCs are usually described as a network of stocks, as in *multi-echelon inventory systems* (Tayur *et al.*, 1998). However, most existing models can only describe a restricted class of SCs, with simplifications. For instance, multi-echelon inventory models focus on inbound logistics (replenishment) or outbound logistics (distribution) with only two echelons, for instance one warehouse and one retailer (Haji *et al.*, 2009; Al-Rifai and Rossetti, 2007). Although these models are more and more relevant, they do not explicitly take account of transportation stages and capacity constraints in SCs: they simply assume a constant lead time between two adjacent stocking locations (Graves *et al.*, 1993; Tayur *et al.*, 1998). These models lack flexibility and generality in describing real-life SCs. Furthermore; no existing mathematical model can describe material, information, and financial flows in an integrated way.

Petri nets are powerful tools for modelling and analysing discrete-event systems such as manufacturing systems (Lindemann, 1998). Since SCs are also discrete-event systems from a high-level of abstraction, tools based on Petri nets have been developed for SC modelling and simulation (Viswanadham and Srinivasa Raghavan, 2000; Chen *et al.*, 2005). A review of the literature about modelling and performance evaluation of inventory system using Petri nets can be found in Labadi *et al.* (2007).

For large-scale complex systems such as SCs, it is very difficult to develop analytical methods for performance evaluation and optimization. The work of Clark and Scarf (1960) is one of the earliest efforts to establish the optimality of echelon stock policies in a serial system operating under periodic review. They used *recursive decomposition* to characterize the optimal echelon base-stock policy and obtain associated inventory costs. However, it is not an easy task as the problem and solution technique are computationally hard to implement.

As an alternative, many researchers have been seeking for *simulation-based optimization methods*, which combine optimization techniques with performance evaluation by simulation (Coello, 2000). The optimization techniques used in these methods include meta-heuristics (Talbi, 2009) such as tabu search and stochastic optimization methods including genetic algorithms (GAs) and simulated annealing. By appropriately combining these meta-heuristics with simulation, near-optimal solutions to stochastic optimization problems can be found in a reasonable computation time. For example, Lee and Billington (1993) developed an SC model operating under a periodic-review base-stock system at Hewlett Packard and applied a search heuristic to find the optimal inventory levels across the SC. Rao *et al.* (2000) developed for Caterpillar an integrated model to analyse different SC configurations for a new line of compact construction equipment, using decomposition techniques, network optimization and applied simulation-based methods (infinitesimal perturbation analysis) to determine inventory levels. Similarly, Ettl *et al.* (2000) analysed a supply network in order to minimize the overall inventory capital throughout the network and to guarantee customer service requirements. They compared a conjugate gradient routine search with the best solution obtained by generating 500,000 random solutions.

In real SCs, several objectives are often considered, for instance the total inventory cost and the service level offered to customers. Such problems are difficult to solve because the objectives are often conflicting and decision makers have different interpretations of optimality when facing several criteria. *Optimization in the Pareto-sense* is probably the most widely accepted interpretation. For two criteria f_1, f_2 to be minimized, one solution s is said to *dominate* another solution t if it is better for one criterion at least, i.e., if and only if $f_1(s) \leq f_1(t)$ and $f_2(s) < f_2(t)$ or $f_1(s) < f_1(t)$ and $f_2(s) \leq f_2(t)$. A solution is *non-dominated* or *efficient* if no other solution dominates it. Solving a multi-objective optimization problem in the Pareto sense consists in finding the set of non-dominated solutions, called the *Pareto-optimal set*. Two efficient solutions s and t can be considered as optimal because moving from s to t improves one criterion but degrades the other.

Finding a set of Pareto-optimal solutions or at least a good approximation requires a robust and efficient method that can efficiently search the entire solution space of the problem. GAs and especially *multi-objective genetic algorithms* (MOGAs) seem to be well suited for this task because they process multiple solutions in parallel and generate improved solutions by combining the good patterns of parent solutions. Moreover, some theoretical results concerning convergence were proved for some multi-objective EAs (Laumann *et al.*, 2002) and more and more robust methods have been proposed in recent years (see for instance Burke and Landa Silva, 2006). According to the studies which have compared several multi-objective algorithms, NSGA-II (Deb *et al.*, 2000) and SPEA-II (Zitzler *et al.*, 2001) are probably the most efficient methods in terms of convergence and diversity of solutions.

The use of EAs in the determination of an SC inventory policy was introduced by Daniel and Rajendran (2005), who proposed a GA to optimize the base-stock levels with the objective of minimizing the sum of holding and shortage costs in the entire SC. Köchel and Nieländer (2005) used a simulation-based optimization method combined with a GA in order to find optimal order-points and order-quantities for all stages in the SC to minimize the total cost per year.

Daniel and Rajendran (2006) were the first to consider two objectives: they studied the minimization of total inventory costs and total shortage costs by a multi-objective optimization GA they called MOGASCIP. Ding *et al.* (2006) developed a simulation-based optimization module named ONE to support decision makers for the assessment, design and improvement of SC networks. Among strategic and tactical issues, inventory stock-levels were included in the search of the optimal Pareto front by the NSGA-II. Finally, in Amodeo *et al.* (2008), the performance of an SC is evaluated by simulating its Petri net model and, again, NSGA-II is used to guide the optimization search process toward high-quality solutions of inventory policies.

To the best of our knowledge, no study has been published to find the best evolutionary technique for a simulation-based multi-objective optimization in SC inventory policy.

9.3 Problem Formulation

An SC is a serial linkage of interrelated echelons, which perform a specific task leading to the final product, see Figure 9.1. The interconnection can be complex and one echelon can play several roles. Generally, *suppliers* feed *manufacturers* with raw materials for production to be carried out. Production is transferred by *distribution centres* to final *retailers* who are responsible for selling the product to a customer.

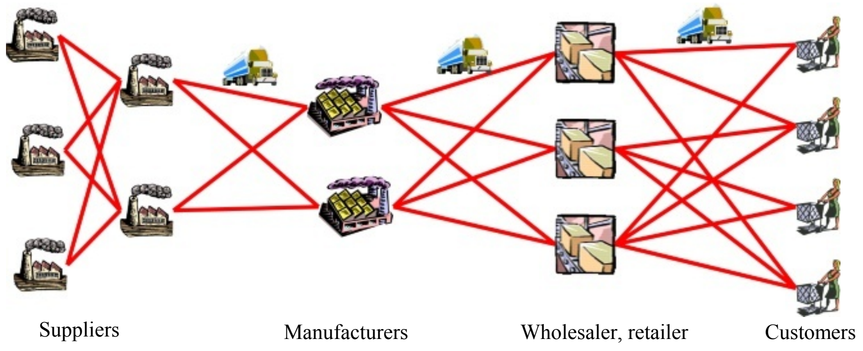


Fig. 9.1 Example of SC network

In order to get the material transferred between echelons, each production facility replenishes the subsequent inventory and places orders on the preceding inventory in the SC. However, if the order exceeds the stock of the preceding inventory, the order is partially filled and the unmet quantity is backordered until it is available in stock. This process can be successfully modelled with discrete-event simulation that takes stochastic demands, production rates and lead times into account.

On the other hand the suggestion of a specific *inventory policy* is achieved by the optimization module. Thus, given a specific SC design (sequential links between different sourcing, production and distribution activities or processes) the attention is focused on how the individual inventory policies can be defined in response to SC performance.

For the purpose of our study, inventory revision is done periodically. This means that every day, if inventory level is below a predetermined order-point (R), an order of size Q is placed in the preceding echelon. Thus, inventory policy is defined when minimum inventory to be held to avoid stockouts (R) and quantity to order every time (Q) is defined for every echelon. Therefore, for an n-facility SC, the decision variables for the optimization procedure can be defined by an *order-point vector* $R = (R_1, R_2, \ldots, R_n)$ and an *order-quantity vector* $Q = (Q_1, Q_2, \ldots, Q_n)$.

For two given vectors R and Q two performance measures are computed by simulation: the *total inventory cost* f_1, to be minimized, and the *customer service level* f_2, to be maximized. Service level is defined as the probability that customer orders can be satisfied on time. It is obtained by computing the fraction of time when the final retailer has orders but there is no inventory available to process them. Inventory cost is calculated by taking into account the annual holding costs of raw materials and final products in stocks.

In this chapter, five different SC configurations are considered to check the robustness of the proposed approach (Figures 9.2, 9.3). The simulation module presented in Amodeo *et al.* (2008) is used to compute the objective functions. For each configuration, the trade-off between customer satisfaction and inventory holding cost is posed as a multi-stage stochastic optimization problem where order-quantities (Q) and reorder-points (R) are the optimization variables. The goal is to compute a set of non-dominated or Pareto-optimal solutions (Zitzler and Thiele, 1999). For our two objectives, a solution i dominates a solution j $(i \prec j)$ if $(f_1(i) \leq f_1(j)$ and $f_2(i) > f_2(j))$ or $(f_1(i) < f_1(j))$ and $(f_2(i) \geq f_2(j))$.

Configuration 1

Composed of three suppliers ($S1$, $S2$, $S3$), a manufacturer ($S4$), and a set of customers. Orders sent by the set of customers are received randomly with exponentially distributed inter-arrival times with mean 0.036 days. Manufacturer needs three raw materials for production purchased from suppliers 1, 2 and 3 respectively. Supply source of raw materials is infinite. For each stock $S1$ to $S4$, there is a lead time (including processing and transportation), equal to 40, 20, 70 and 20 days respectively. Similarly, the annual holding costs of raw materials and final products in stocks $S1$, $S2$, $S3$, $S4$ are 0.3€, 0.6€, 0.16€ and 3€.

Configuration 2

Echelon links and parameters remain as in configuration 1, but demand orders are received randomly according to normally distributed inter-arrival times with mean 0.04 days and variance 0.002 days.

Configuration 3

Echelon links and demand arrival are the same as in configuration 1. However, the means of exponentially distributed lead times change to 50, 30, 80 and 10 days and the annual holding costs become 0.4€, 0.7€, 0.2€ and 2.5€ for the stocks *S*1 to *S*4, respectively.

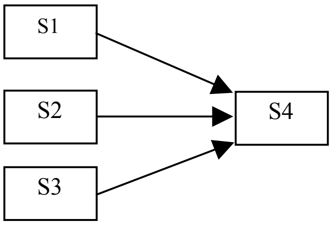


Fig. 9.2 Configurations 1, 2 and 3

Configuration 4

Two echelons *S*5 and *S*6 are added downstream to configuration 1. *S*4 is treated as an intermediate manufacturer. Both *S*5 and *S*6 receive demand orders with exponential arrival times with means 0.05 and 0.04 days. Holding costs in *S*5 and *S*6 are 3€ and 4€ respectively. Processing and transportation lead times from *S*4 to *S*5 and *S*6 are exponential with means 50 and 30 days. *S*5 and *S*6 have processing and transportation lead times of 10 and 15 days.

Configuration 5

A single echelon *S*0 is added as a source to configuration 1 so that *S*1, *S*2, *S*3 are treated as manufacturers. All characteristics of configuration 1 are maintained but production and process lead times from *S*0 to *S*1, *S*2 and *S*3 are now exponentially distributed with means 40, 50 and 45.

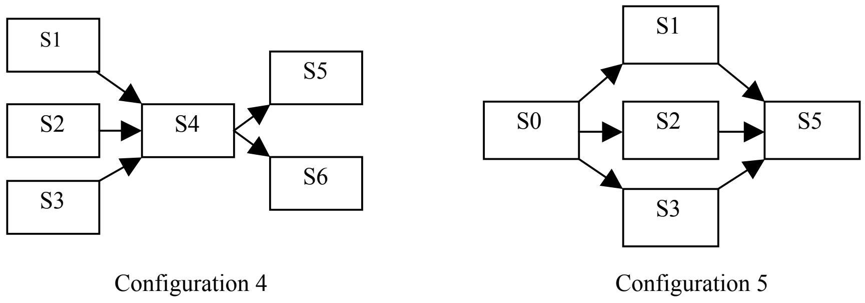


Fig. 9.3 Configurations 4 and 5

9.4 Implementation of Selected Evolutionary Algorithms

This section briefly explains the principles of the EAs selected for the study and how they have been adapted for our SC problem. The first three ones (NSGA-II, SPEA-II and SPEA-IIb) are GAs. GAs are search optimization procedures motivated by the principles of natural selection. They are implemented by creating a population of candidate solutions called *chromosomes*, which in turn are evaluated and selected to evolve toward better solutions, using crossover and mutation operators. The fourth algorithm is MOPSO, in which the population moves in the solution space instead of generating offspring via genetic operators.

9.4.1 Non-dominated Sorting Genetic Algorithm II

NSGA-II computes successive generations on a population of solutions kept sorted by non-domination. A *non-dominated sorting* starts by computing the set of non-dominated solutions of the population, called front 1. These solutions are virtually removed and the non-dominated solutions in the residual population are determined to get front 2. This process is repeated until all individuals are assigned to a front, see Figure 9.4a.

A *crowding distance* is used to control population diversity. The computation of this distance for one solution is easy to explain for our two objectives. Assume that the front F which contains the solution (after the non-dominated sorting) is sorted in ascending order of objective f_1. Let i be the rank of the solution in the sorted front F and d_i the crowding distance to be computed. If i is the first or last solution in F ($i = 1$ or $i = |F|$), then $d_i = \infty$, otherwise:

$$d_i = \frac{f_1(i+1) - f_1(i-1)}{f_1^{\max} - f_1^{\min}} + \frac{f_2(i+1) - f_2(i-1)}{f_2^{\max} - f_2^{\min}} \tag{9.1}$$

where $i-1$ and $i+1$ are the solutions before and after i and $f_k^{\max}$ and $f_k^{\min}$ the minimum and maximum values of objective k.

The crowding distance is illustrated by Figure 9.4b. It is the half-perimeter of the rectangle whose two opposite corners correspond to the predecessor I-1 and to the successor i+1 of solution i in its front F (this rectangle is called *cuboid* by Deb *et al.* (2000). Clearly, the purpose of this distance is to favour the most isolated solutions and to avoid clusters of very similar solutions.

In NSGA-II, the selection of two parents p_1 and p_2 for reproduction is realized by a *crowded tournament operator*. Let rank(i) be the rank of a solution i in its front. To get the first parent p_1, two solutions a and b are randomly selected. If rank(a) < rank(b) or rank(a) = rank(b) and $d_a > d_b$, then $p_1 = a$, otherwise $p_1 = b$. The same process is repeated to get the other parent. In other words, NSGA-II uses the front number of a solution as fitness and, when two solutions belong to the same front, ties are broken using their crowding distances.

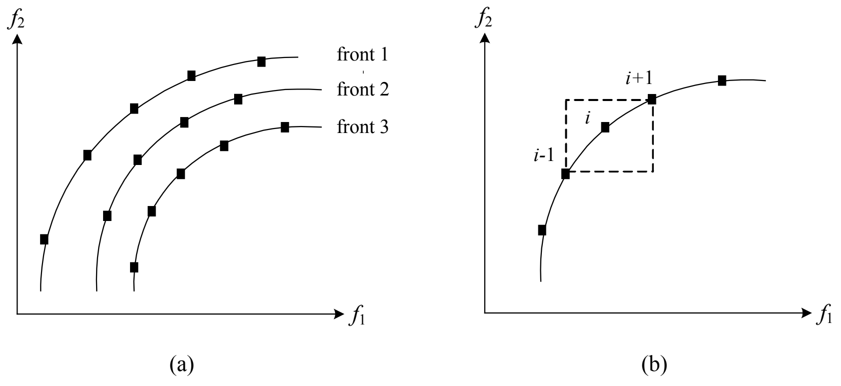


Fig. 9.4 Non-dominated sorting (a) and crowding distance (b)

The overall structure of the NSGAII is specified by the following algorithm. The crossover and mutation operators are the same as for SPEA-II, see Sect. 9.4.2.

NSGA-II Procedure

1. Initialization: generate an initial population P_0 of N chromosomes; evaluate them using the simulator; set generation counter $t = 0$.
2. Sort P_0 by non-domination and compute the crowding distance of each solution.
3. Perform N/2 reproductions: for each reproduction, select two parents p_1 and p_2 using the crowded tournament operator, and apply crossover and mutation. This gives N children-solutions which are added at the end of P_t.

4. Sort P_t by non-domination and compute the crowding distance of each solution.
5. Truncate: initialize an auxiliary population $P_{\text{new}} = \varnothing$. Copy the successive fronts of P_t into P_{new} until P_{new} gets full or the current front F cannot be copied entirely. In the latter case, copy the solutions of F in decreasing order of crowding distance, until P_{new} contains exactly N solutions.
6. Termination: increment generation counter $(t = t + 1)$; if $t \geq T$ (maximum number of generations), return the non-dominated individuals in P_t and stop.
7. Set $P_t = P_{\text{new}}$ and go to step 3.

9.4.2 Strength Pareto Evolutionary Algorithm II

SPEA-II was proposed by Zitzler *et al.* (2001) as an improved version of SPEA (Zitzler and Thiele, 1999). Contrary to NSGA-II, SPEA-II does not use a non-dominated sorting: the fitness of a solution is the number of solutions it dominates and the best non-dominated solutions are recorded in an auxiliary population called an *archive*.

Chromosome Representation

A *phenotype* representation is used as one works directly with the actual values of decision variables. One chromosome contains two substrings: $[R_1, R_2, ..., R_n]$ for the order-point vector (R) and $[Q_1, Q_2, ..., Q_n]$ for the order-quantity vector (Q).

SPEA-II Procedure

1. Initialization: generate an initial population P_0 of N chromosomes; prepare an empty external archive $\overline{P_0} = \varnothing$; set generation counter $t = 0$.
2. Fitness assignment: calculate fitness values for individuals in P_t and $\overline{P_t}$.
3. Environmental selection: copy non-dominated individuals of $P_t \cup \overline{P_t}$ to $\overline{P_{t+1}}$. If $|\overline{P_{t+1}}| > \overline{N}$ (archive size), then reduce $\overline{P_{t+1}}$ using a truncation operator, else, if $|\overline{P_{t+1}}| < \overline{N}$, fill $\overline{P_{t+1}}$ with dominated individuals present in P_t and $\overline{P_t}$.
4. Termination: if $t \geq T$ (maximum number of generations), return the non-dominated individuals in $\overline{P_{t+1}}$, stop. Otherwise, go to next step.
5. Mating selection: perform *binary tournament selection* with replacement on $\overline{P_t}$ to fill the mating pool.

6. Variation: apply recombination and mutation operators to the mating pool and set P_{t+1} to resulting population. Increment generation counter ($t = t+1$) and go to step 2.

- *Initialization*: for each network site i, we are given lower and upper values $R_i^{\min}$, $R_i^{\max}$, $Q_i^{\min}$, $Q_i^{\max}$ for R_i and Q_i. Initial individuals are randomly generated as follows: $R_i = U \sim [R_i^{\min}, R_i^{\max}]$ and $Q_i = U \sim [Q_i^{\min}, Q_i^{\max}]$, $\forall\ i = 1, 2, \ldots, n$.
- *Fitness assignment*: each individual i in archive $\overline{P_t}$ and population P_t is assigned a strength value $S(i)$ which represents the number of solutions it dominates, i.e., $S(i) = |\{j: j \in \overline{P_t} \cup P_t \wedge i \succ j\}|$. The fitness $F(i) = Z(i)+D(i)$ includes a raw fitness $Z(i) = \sum_{j \in \overline{P_t}+P_t \wedge j \succ i} S(j)$ and a term $D(i)$ representing the closeness of solutions. $D(i) = 1/(\sigma_i^k + 2)$ where σ_i^k is the Euclidean distance between i and its k-th nearest neighbor, with $k = \sqrt{N}$.
- *Chromosome selection*: chromosomes are copied according to their fitness $F(i)$ into a mating pool. Binary tournament selection is used. Two individuals are randomly selected from population P_t and the one with the best fitness (smallest $F(i)$) is copied to the pool. This process is repeated until the mating pool is full.
- *Crossover and mutation*: a single-point crossover is utilized. A pair of children is created by exchanging the halves of the two parents delimited by one randomly selected cutting point. For mutation, the *gene-wise mutation* of Daniel and Rajendran (2005) is used. Each chromosome undergoes mutation with a fixed small probability and each of its genes i is modified as follows: $R'_i = \min \{\max\{R_i(1-x) + 2xu \times R_i,\ R_i^{\min}\ ,\ R_i^{\max}\}\}$ where $x \in (0,1)$ is a constant determined at the beginning and $u \sim U[0,1]$ is a uniform random number.
- *Truncation*: for archive truncation, the solution with the smallest closeness value for $k = 1$ is eliminated. If two solutions share the same closeness for $k = 1$, the values $k = 2$ onwards are successively tested until these values differ.

9.4.3 Strength Pareto Evolutionary Algorithm IIb

We modified SPEA-II to obtain a new version, SPEA-IIb, aimed at taking advantage of the information gained by the simulator by introducing an aggressive reproduction scheme. Another purpose was to improve the uniformity of the final Pareto front by changing the original truncation operator. The general architecture of SPEA-II algorithm is maintained, and the only two changes are:

- *Crossover*: in SPEA-II, the *Variant II: gene-wise* reproduction technique proposed by Daniel and Rajendran (2005) is utilized. Here, for one offspring, the information of as many parents as inventories in the SC is utilized. In our case, four parents are chosen at random and one offspring is created. The process is continued until the offspring pool is filled.

- *Archive Truncation*: the crowding distance of Deb *et al.* (2000), already explained for NSGA-II, is used to truncate the archive. The crowding distance d_i of each solution in the external archive $\overline{P_{t+1}}$ is calculated and the individual with smallest crowding distance is eliminated from the archive.

9.4.4 Multi-objective Particle Swarm Optimization

PSO is inspired by the flocking behavior of birds and was first proposed by Kennedy and Eberhart (1995). PSO can be viewed as a population-based heuristic: the population, called, a *swarm*, contains a set of individuals (solutions) called *particles*, which fly progressively toward an optimum solution in the search space. Several adaptations of the original PSO have been made to tackle problems with multiple objectives. We took the very recent implementation of Tripathi *et al.* (2007) as it introduces a novel concept of time-varying coefficients, while looking effective and easily adaptable to our problem. Each particle i is defined by the set of following attributes:

- *Position attribute*: in the original PSO, this attribute is defined by the position $X_i = (x_{i1}, x_{i2}, \ldots, x_{id})$ of particle i in the d-dimensional search space. In our supply chain problem, $d = 2n$, x_{i1} to x_{in} correspond to the order-points R_1 to R_n and $x_{i,n+1}$ to $x_{i,2n}$ to the order-quantities Q_1 to Q_n.
- *Velocity attribute*: this is a vector $V_i = (v_{i1}, v_{i2}, \ldots, v_{id})$ used for computing the displacement in each of the dimensions ruled by the position attribute.
- *Individual experience attribute*: this vector $P_i = (p_{i1}, p_{i2}, \ldots, p_{id})$ records the position corresponding to the best performance of the particle. Similarly, the experience of the whole swarm is expressed by its global best experience, defined by the index g of the particle which found the best solution: in other words, this is the particle with the best vector P.

The movement of a particle i toward the optimum is governed by updating its position and velocity attributes at each iteration (see Tripathi *et al.*, 2007):

$$x_{ij} = x_{ij} + v_{ij}\,,$$

$$v_{ij} = w \cdot v_{ij} + c_1 \cdot r_1 \cdot (p_{ij} - x_{ij}) + c_2 \cdot r_2 \cdot (p_{gj} - x_{ij})\,, \tag{9.2}$$

where $w \geq 0$ is the *inertia weight*, c_1 and c_2 are non-negative *acceleration coefficients*, and r_1, $r_2 \in [0,1]$ are random numbers which bring some randomness to the flight. The *cognition term* $c_1 \cdot r_1 \cdot (p_{ij} - x_{ij})$ reflects the individual experience of the particle, while the *social term* $c_2 \cdot r_2 \cdot (p_{gj} - x_{ij})$ models a social interaction between particles. Increasing the *cognitive acceleration coefficient* c_1 favours exploration, while augmenting the *social acceleration coefficient* c_2 stimulates exploitation.

Compared with the original PSO, MOPSO uses an archive containing the best non-dominated solutions found since the beginning of the flight. The individual experience attribute of a particle is initialized with its initial position and updated each time a new position dominates it. The global best particle is a non-dominated solution selected at each iteration in the archive, using a density measure based on the distance to the nearest neighbour in the archive. The global best solution is randomly picked using a roulette wheel selection which favours high densities. So, the global best solution is in general a relatively isolated non-dominated solution.

Tripathi *et al.* (2007) proposed time-varying inertia and acceleration coefficients $w(t)$, $c_1(t)$ and $c_2(t)$ to be used at iteration t of MOPSO to have an adequate balance between local exploitation and global exploration. To ensure smaller moves at the end, the inertia coefficient $w(t)$ is allowed to decrease linearly between an initial value w_i and a final value w_f, while the cognitive (local) coefficient $c_1(t)$ reduces from $c_{1\mathrm{i}}$ to $c_{1\mathrm{f}}$. To favour the convergence to the present global best at the end, the social coefficient $c_2(t)$ is increased from $c_{2\mathrm{i}}$ to $c_{2\mathrm{f}}$.

Hence, the coefficients are updated as follows, where T is the maximum number of iterations:

$$c_1(t) = c_1(t) - \frac{t}{T}(c_{2i} - c_{2f}), c_2(t) = c_2(t) + \frac{t}{T}(c_{2f} - c_{2i}), w(t) = w(t) - \frac{t}{T}(w_i - w_f) \tag{9.3}$$

PSO is known for its good convergence but this convergence is usually achieved at the cost of diversity in the multi-objective case. Tripathi *et al.* (2007) suggest the inclusion of mutation to remedy this problem. In our case, we selected the *gene-wise* mutation (Deb *et al.*, 2000) already presented in Sect. 9.4.2. Using all these ingredients, MOPSO can be sketched by the following algorithm.

MOPSO Procedure

1. *Initialization*: generate one initial swarm F_0 of N particles with inventory policies randomly generated like in SPEA-II and null velocities; put in the external archive $\overline{F_0}$ the non-dominated solutions of F_0; set generation counter $t = 0$.
2. *Update global best performance*: update the global best particle index g.
3. *Update personal best*: update the experience attribute P_i of each particle i.
4. *Adjust coefficients* $c_1(t)$, $c_2(t)$ and $w(t)$.
5. *Update velocities and positions*: compute the new velocity attribute V_i and the new position X_i of each particle i.

6. *Update archive*: copy all non-dominated individuals in $F_t \cup \overline{F_t}$ to $\overline{F_{t+1}}$. If $|\overline{F_{t+1}}| > \overline{N}$ (archive size), keep only the $\overline{N}$ most sparsely spread solutions using the density measure.
7. *Termination*: if $t \geq T$ (maximum number of iterations) then return the non-dominated individuals in archive $\overline{F_{t+1}}$ and stop. Otherwise, go to next step.
8. *Mutation*: apply the mutation operator to F_t and set F_{t+1} to the resulting population. Increment iteration counter ($t = t+1$) and go to 2.

9.5 Computational Experiments and Analysis

9.5.1 Evaluation Criteria

When evaluating a multi-objective optimization technique, two complementary characteristics are sought: the solutions must be as close as possible to the Pareto-optimal set (*closeness*) and as diverse as possible in the non-dominated front *(diversity*). Accordingly, three evaluation criteria are used:

1. The *coverage of two sets* $C(X,Y)$ proposed by Zitzler and Thiele (1999). For two sets of solutions X and Y, it is defined as the fraction of solutions in Y which are dominated (covered) by at least one solution in X. For instance, $C(X,Y) = 1$ means that all points in Y are dominated by points in X. Both $C(X,Y)$ and $C(Y,X)$ must be computed as C is not symmetric.

$$C(X,Y) = \frac{|\{y \in Y : \exists\, x \in X : x \succ y\}|}{|Y|}. \tag{9.4}$$

2. *Spacing*. Schott (1995) suggested a relative distance measure S between solutions in a non-dominated set Q, a small value indicating a more uniform spacing:

$$S = \sqrt{\frac{1}{Q}\sum_{i=1}^{|Q|}\left(d_i - \overline{d}\right)^2} \tag{9.5}$$

where $\overline{d}$ is the average of the d_i, the distance between solution i and the nearest solution in objective space. Each d_i is computed as below for M objectives:

$$d_i = \min\left\{ \sum_{m=1}^{M} (f_m^i - f_m^k)^2 \mid k \in Q \wedge k \neq i \right\}. \tag{9.6}$$

9.5.2 Parameters Used

All algorithms were programmed on a 2.4 GHz Windows workstation using C++ under the CodeBlocks environment. Simulation parameters are the following: simulation horizon of 2000 days, 25 replications, warm-up of 20 days. 1000 days were enough for the first three configurations but the two last required 2000 days due to additional echelons. The average time per call to the simulator was 0.79 s for configurations 1 to 3, 4.13 s for configuration 4 and 3.98 s for configuration 5. SPEA-II, SPEA-IIb, MOPSO and NSGA-II were tested for the five SC configurations during 1000 generations, using the additional parameters of Table 9.1. NSG-II is the only algorithm without archive. To have a fair comparison with the 50 solutions returned by SPEA-II, SPEA-IIb and MOPSO, we selected at the end of NSGA-II the 50 non-dominated solutions with the larger crowding distance.

Table 9.1 Algorithm specifications

Parameter	SPEA-II	SPEA-IIb	NSGA-II	MOPSO
Mutation probability	0.1	0.4	0.01	0.1
Crossover probability	0.9	0.9	0.9	–
Archive size	50	50	50	50
Population size	100	100	100	100
Inertia coefficient	–	–	–	$0.7 \rightarrow 0.4$
Cognitive coefficient	–	–	–	$2.5 \rightarrow 0.5$
Social coefficient	–	–	–	$0.5 \rightarrow 2.5$

9.5.3 Experimental Results

Table 9.2 gives the running times in seconds. The spacing measures for the non-dominated solutions returned by each algorithm are listed in Table 9.3. Tables 9.4 and 9.5 provide the values of Zitzler's measure, using NSGA-II as reference.

Table 9.2 Computational time in seconds

Configuration	SPEA-II	SPEA-IIb	NSGA-II	MOPSO
1	17080	15849	55762	17485
2	17125	15909	55836	17497
3	17369	16239	55781	17806
4	22940	20732	131797	23765
5	22943	20746	131875	23802

Table 9.3 Spacing measure

Configuration	SPEA-II	SPEA-IIb	NSGA-II	MOPSO
1	1.5500	1.4030	0.7900	1.4110
2	1.5944	1.4335	0.7997	1.4464
3	1.6016	1.4705	0.8013	1.4557
4	2.5650	2.0623	0.8940	2.1191
5	2.5746	1.9901	0.9237	2.1768

Table 9.4 Zitzler's measure in %, C (NSGA-II, other algorithm)

Configuration	1	2	3	4	5
SPEA-II	32	88	84	4	0
SPEA-IIB	0	2	0	0	0
MOPSO	58	100	100	96	58

Table 9.5 Zitzler's measure in %, C (other algorithm, NSGA-II)

Configuration	1	2	3	4	5
SPEA-II	26	14	8	46	66
SPEA-IIB	100	96	100	100	100
MOPSO	10	0	0	8	26

Figures 9.5 to 9.9 compare the final pareto fronts obtained by the four methods, on the five configurations. Some important observations can be deduced:

1. Our version SPEA-IIb of SPEA-II is superior for all configurations tested. No point of the SPEA-IIb front is dominated by NSGA-II solutions. The only exception is configuration 2, in which one point is dominated.
2. MOPSO is outperformed by the three true EAs, which provide substantially better fronts according to Zitzler's measure. This weakness is particularly visible on configurations 2 to 5 (see Figures 9.6 to 9.9).
3. There is not a clear second place in terms of domination between NSGA-II and SPEA-II. For some configurations, more individuals in NSGA-II dominate the ones of SPEA-II while for other configurations the opposite occurs. However, for the two most complicated configurations, 4 and 5, SPEA-II is superior.
4. NSGA-II spacing is the most uniform on all configurations. This interesting property probably results from the crowding distance used by this algorithm.
5. Although some minor changes were made to the original algorithm SPEA-II, SPEA-IIb gives better results with its different crossover and archive truncation method. In particular, Schott's spacing measure is improved using the crowding distance from NSGA-II in archive truncation.
6. In spite of NSGA-II superior spacing, this algorithm takes longer than the other methods and its running time explodes for configurations 4 and 5.
7. All algorithms were able to find 50 non-dominated solutions: on all instances, SPEA-II, SPEA-IIb and MOPSO returned an archive filled by non-dominated solutions while NSGA-II yielded at least 50 solutions.

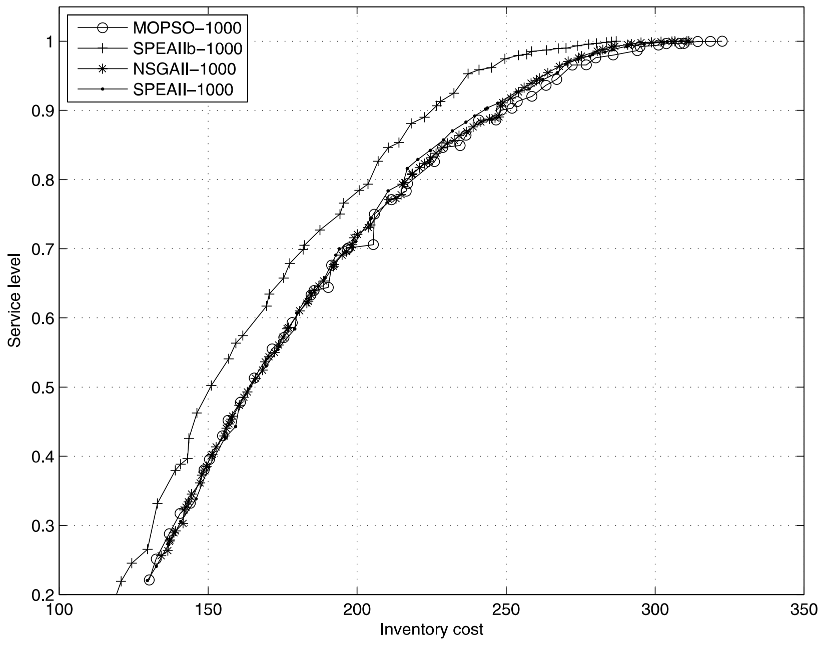


Fig. 9.5 Final Pareto front for each algorithm on configuration 1

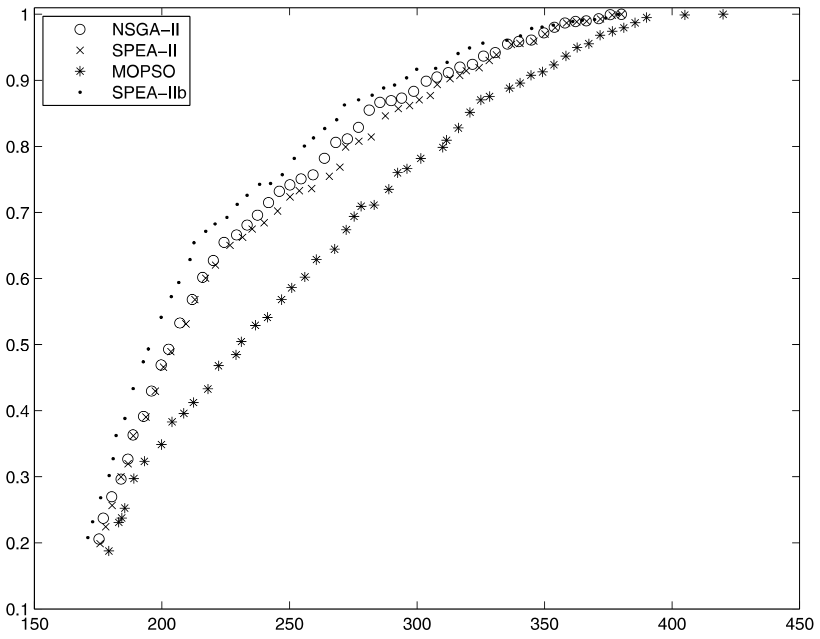


Fig. 9.6 Final Pareto front for each algorithm on configuration 2

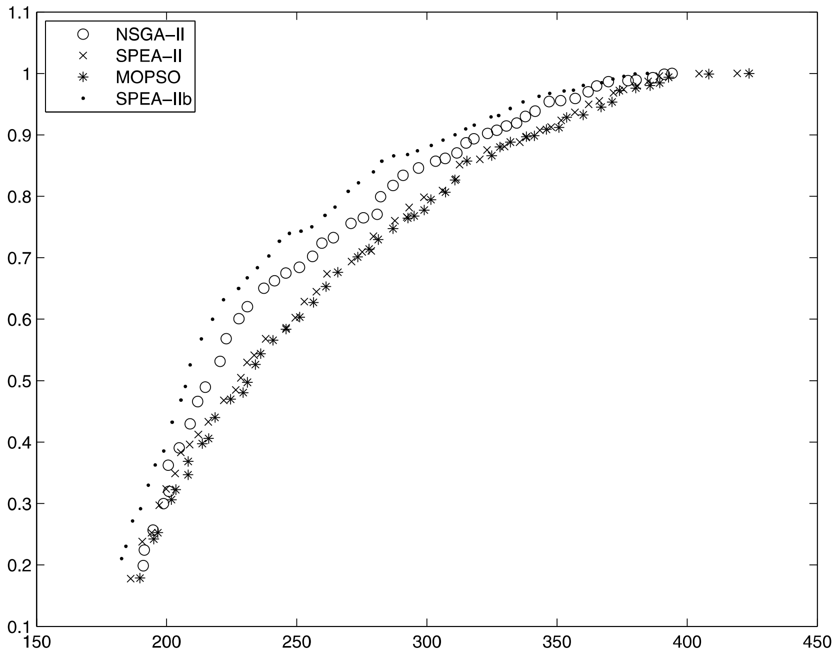


Fig. 9.7 Final Pareto front for each algorithm on configuration 3

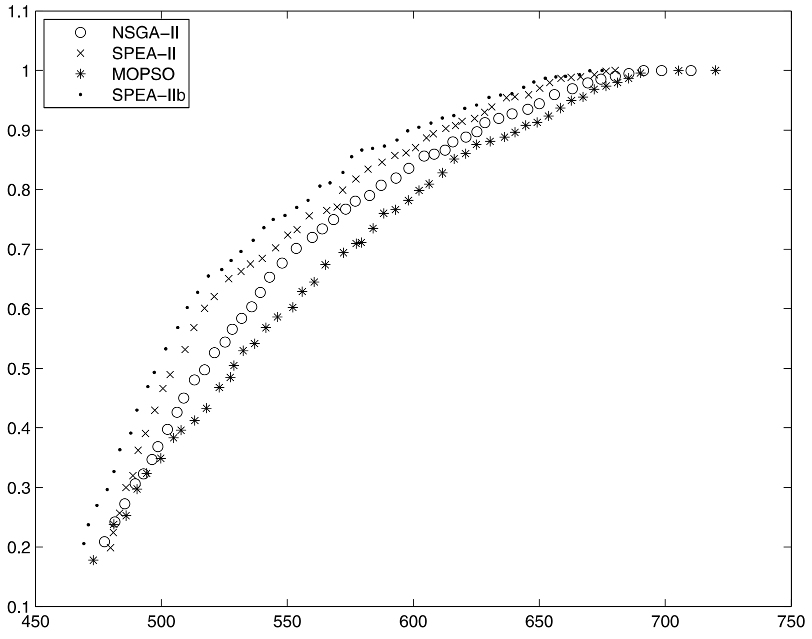


Fig. 9.8 Final Pareto front for each algorithm on configuration 4

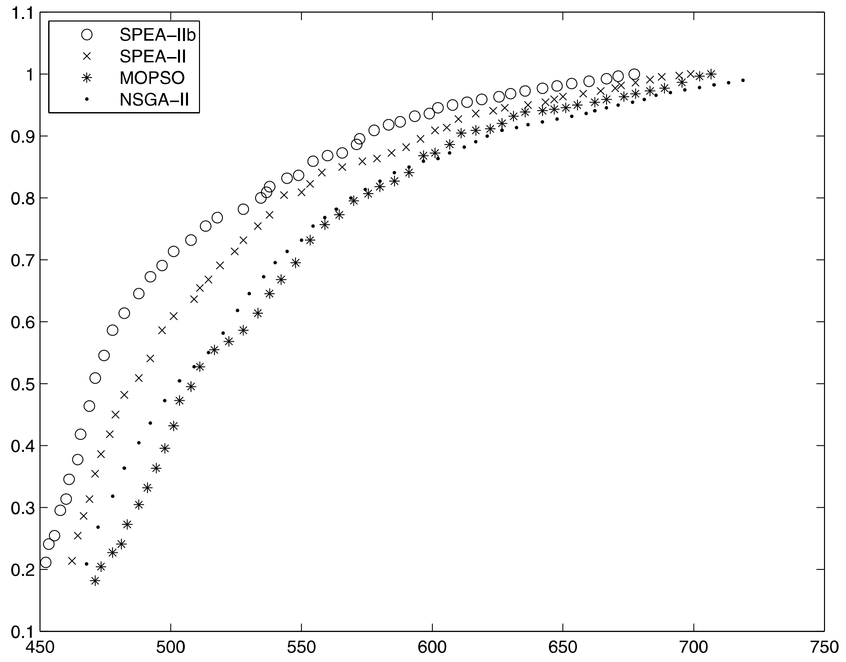


Fig. 9.9 Final Pareto front for each algorithm on configuration 5

9.6 Conclusion

The purpose of this research was to determine which EA is best suited to be coupled with an SC simulation module in solving a bi-objective SC optimization problem. Four different algorithms were tested on five different configurations and compared in terms of efficiency for reaching the optimal Pareto front. The results indicate that the SPEA-II architecture makes the best use of the simulator and that crossover and spacing techniques are critical, as shown by the better variant SPEA-IIb. Our tests have even shown that the convergence of SPEA-IIb is faster during the first hundred generations.

On the other hand, MOPSO is still far from other techniques and should be improved to exploit its full potential. Further exploration on reproduction methods could further improve results. NSGA-II spacing power could be integrated with SPEA-II architecture in order to reach optimization potential.

References

Al-Rifai MH, Rossetti MD (2007) An efficient heuristic optimization algorithm for a two-echelon (R, Q) inventory system. Int. J. Prod. Eco. 109(1–2):195–213

Amodeo L, Chen H, El Hadji A (2008) Supply chain inventory optimization with multiple objectives: An industrial case study. In: Fink A, Rothlauf F (eds): Advances in computational intelligence in transport, logistics, and supply chain management, Springer-Verlag, 211–230

Burke EK, Landa Silva JD (2006) The influence of the fitness evaluation method on the performance of multiobjective search algorithms. Eur. J. Oper. Res. 169(3):875–897

Chen H, Amodeo L, Chu F. et al. (2005) Modelling and performance evaluation of supply chains using batch deterministic and stochastic petri nets. IEEE Trans. Auto. Sci. Eng. 2(2):132–144

Clark AJ, Scarf H (1960) Optimal policies for a multi-echelon inventory problem. Mgmt. Sci. 6:475–490

Coello CA (2000) An updated survey of GA-based multiobjective optimization techniques. ACM Comp. Surveys, 32(2):109–143

Daniel JS, Rajendran C (2005) A simulation-based genetic algorithm for inventory optimization in a serial supply chain. Int. Trans. Oper. Res. 12:101–127

Daniel JS, Rajendran C (2006) Heuristic approaches to determine base-stock levels in a serial supply chain with a single objective and multiple objectives. Eur. J. Oper. Res. 175:566–592

Deb K, Agrawal S, Pratab A et al. (2000) A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: Proceedings of Parallel Problem Solving From Nature (PPSN VI), Paris, France, pp. 849–858

Ding H, Benyoucef L, Xie X (2006) A simulation-based multi-objective genetic algorithm approach for networked enterprises optimization. Eng. App. Art. Intell. 19:609–623

Ettl M, Feign GE, Lin GY et al. (2000) A supply network model with base-stock control and service requirements. Oper. Res. 48:216–232

Fu MC (2001) Simulation optimization. In: Proceedings of the 2001 Winter Simulation Conference, pp. 53-61

Graves SC, Rinnooy Kan AHG, Zipkin P (1993) Handbooks in operations research and management science: logistics of production and inventory. Elsevier Science Publisher

Haji R, Pirayesh Neghab M, Baboli A (2009) Introducing a new ordering policy in a two-echelon inventory system with Poisson demand. Int. J. Prod. Eco. 117(1):212–218

Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of the IEEE Int. Conf. on Neural Networks, pp. 1942–1948, Piscataway, NJ

Köchel P, Niëlander U (2005) Simulation-based optimization of multi-echelon inventory systems. Int. J. Prod. Eco. 93–94:505–513

Labadi K, Chen H, Amodeo L (2007) Modeling and performance evaluation of inventory systems using batch deterministic and stochastic petri nets. IEEE Trans. Syst. Man Cyber.–Part C: Applications and Reviews 37(6):1287–1302

Laumanns M, Thiele L, Deb K et al. (2002) Combining convergence and diversity in evolutionary multiobjective optimization. Evol. Comp. 10(3):263–282

Lee HL, Billington C (1993) Material management in decentralized supply chains. Oper. Res. 41:835–847

Lindemann C (1998) Performance modeling with deterministic and stochastic petri nets. John Wiley and Sons

Rao U, Scheller-Wolf A, Taytur S (2000) Development of a rapid-response supply chain at Caterpillar. Oper. Res. 48:189–204

Schott JR (1995) Fault tolerant design using simple and multi-criteria genetic algorithms. Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Boston, MA, USA

Talbi E (2009) Metaheuristics – from design to implementation. John Wiley and Sons

Tayur S, Ganeshan R, Magazine M (1998) Quantitative models for supply chain management. Kluwer Academic Publishers

Tripathi PK, Bandyopadhyay S, Pal SK (2007) Multi-objective particle swarm optimization with time variant inertia and acceleration coefficients. Inf. Sci. 177:5033–5049

Viswanadham N, Srinivasa Raghavan NR (2000) Performance analysis and design of supply chains: A petri net approach. J. Oper. Res. Soc. 51(10)

Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. IEEE Trans. Evo. Comp. 3:257–271

Zitzler E, Laumanns M, Thiele L (2001) SPEA2: Improving the strength Pareto evolutionary algorithm. Technical report 103, Gloriastrasse 35, CH-8092 Zurich, Switzerland

Chapter 10
Diverse Risk/Cost Balancing Strategies for Flexible Tool Management in a Supply Network

D. D'Addona and R. Teti

Abstract This work is a part of a wider-scope research concerned with the development and implementation of a multi-agent tool management system (MATMS) for automatic tool procurement. The design, functioning, and performance of diverse flexible tool management strategies integrated in the MATMS is illustrated here. The MATMS operates in the frame of a negotiation based multiple-supplier network where a turbine blade producer (customer) requires from external tool manufacturers (suppliers) the performance of dressing operations on worn-out cubic boron nitride grinding wheels for nickel base alloy turbine blade fabrication. The diverse FTMS paradigms, configured as domain-specific problem-solving functions operating within the MATMS intelligent agent holding the responsibility for optimum tool inventory sizing and control, have been tested by tool inventory management simulations and comparison with real industrial cases.

Keywords Tool management, inventory control, multi-agent systems, supply networks

10.1 Introduction

In recent times, novel software architecture to manage supply networks at the tactical and operational levels has emerged. The supply network is viewed as a system made of a set of intelligent (software) agents, each responsible for one or more activities in the supply network and each interacting with other agents in planning and executing their responsibilities (Fox *et al.*, 2000). The adoption of agent-based or multi-agent technology is founded on the three main system domain characteristics (Yuan *et al.*, 2001): data, control, expertise or resources are

D. D'Addona (✉) and R. Teti
Department of Materials and Production Engineering, University of Naples Federico II, Piazzale Tecchio, Naples, 80125, Italy
e-mail: daddona@unina.it

inherently distributed; the system is naturally regarded as a society of autonomous cooperating components; the system contains legacy components that must interact with other, possibly new software components.

Supply network management by its very nature has all the above domain characteristics (Sycara, 1998). A supply network consists of suppliers, factories, warehouses, etc., working together to fabricate products and deliver them to customers. Parties involved in the supply network have their own resources, capabilities, tasks, and objectives. They cooperate with each other autonomously to serve common goals but also have their own interests. A supply network is dynamic and involves the constant flows of information and materials across multiple functional areas both within and between network members. Multi-agent technology therefore appears to be particularly suitable to support collaboration in supply network management.

This research work is concerned with the development and implementation of a multi-agent tool management system (MATMS)[1] for automatic tool procurement (Teti and D'Addona, 2003a). The MATMS operates in the frame of a negotiation-based multiple-supplier network where a turbine blade producer (customer) requires from external tool manufacturers (suppliers) the performance of dressing operations on worn-out cubic boron nitride (CBN) grinding wheels for nickel base alloy turbine blade fabrication (Teti and D'Addona, 2003a,b).

In this chapter, the design, functioning and performance of diverse flexible tool management strategies (FTMS) integrated in the MATMS are illustrated. Each FTMS paradigm is configured as a domain-specific problem-solving function operating within the intelligent agent of the MATMS, resource agent (RA), holding the responsibility for optimum tool inventory sizing and control.

The remainder of the chapter is organized as follows: Sect. 10.2 presents the functioning of the developed MATMS, Sect. 10.3 discusses diverse FTMS as an alternative to the traditional tool management procedure, Sect. 10.4 illustrates tool inventory management simulations using the implemented FTMS paradigms, Sect. 10.5 presents the test case applications. Sect. 10.6 concludes the chapter with some perspectives for future work.

10.2 Multi-agent Tool Management System

The production of an aircraft engine model involves the machining of turbine blades through a pre-determined set of CBN grinding wheel part numbers (P/N). Each P/N can machine a maximum number of turbine blades (pieces) during its

[1] For further information and details on the custom made proprietary MATMS and FTMS software packages, please contact Prof. Roberto Teti, Director of the Laboratory for Advanced Production Technologies (LAPT), Dept. of Materials and Production Engineering, University of Naples Federico II, 80125 Naples, Italy, email: roberto.teti@unina.it

planned life cycle. Then, it is shipped to an external tool supplier to be dressed and remains out of stock for a time period defined as dressing cycle time.

For each P/N, a sufficient number of CBN grinding wheel serial numbers (S/N) must be in stock at all times (on-hand inventory) to prevent tool run-out and production interruptions. The P/N on-hand inventory size, *I*, is controlled by: number of pieces/month, *P*; number of pieces/wheel, *G*; number of months required without new or dressed wheel supply, *C* (coverage period), heuristically selected.

The wheel demand, *D*, for each P/N is given by:

$$D = (P/G) \times C - I_0 \tag{10.1}$$

where P/G = tool demand rate (number of wheels/month) and I_0 = initial P/N inventory size.

In traditional tool management, the CBN grinding wheel P/N inventory size is strategically planned by selecting, on a heuristic basis, an appropriate coverage on-hand inventory (number of S/N for production needs in the coverage period *C*). This procedure does not always prove adequate: in some cases, the expected inventory level trend matches the required one; in other cases, it is underestimated, with risk of stock-out, or overestimated, with excessive capital investment (Teti and D'Addona, 2003b). Accordingly, the P/N inventory level is increased or reduced on the basis of skilled logistics staff knowledge, resulting in historical inventory size trends: these are fit solutions that prevent tool run-out and useless investment and can be used as a reference for assessing alternative tool management strategies, independent from expert staff performance.

The MATMS activities are performed according to the multi-agent interaction and cooperation protocols illustrated in Figure 10.1 (Teti and D'Addona, 2003a–c). The figure reports the block scheme of the developed MATMS divided into three functional levels:

1. the supplier network level, including the external tool manufacturers in the supply network;
2. the enterprise level, including the logistics of the turbine blade producer company;
3. the plant level, including the production lines of the turbine blade producer company.

The supplier network level is responsible for carrying out the dressing jobs on worn-out CBN grinding wheels. It comprises only one type of agent, the supplier order acquisition agents (SOAA_i), representing the external tool manufacturers (suppliers) activity of acquiring dressing job orders from the turbine blade producer (customer). The SOAA_i negotiate dressing job orders with the enterprise order distribution agent (ODA) and exchange worn-out/dressed CBN grinding wheels with the enterprise warehouse timer agent (WTA).

The enterprise level is responsible for coordinating the MATMS activities to achieve the best possible results in terms of its goals, including on-time delivery, cost minimization, and so forth. It comprises different intelligent agents performing the fundamental tool management activities.

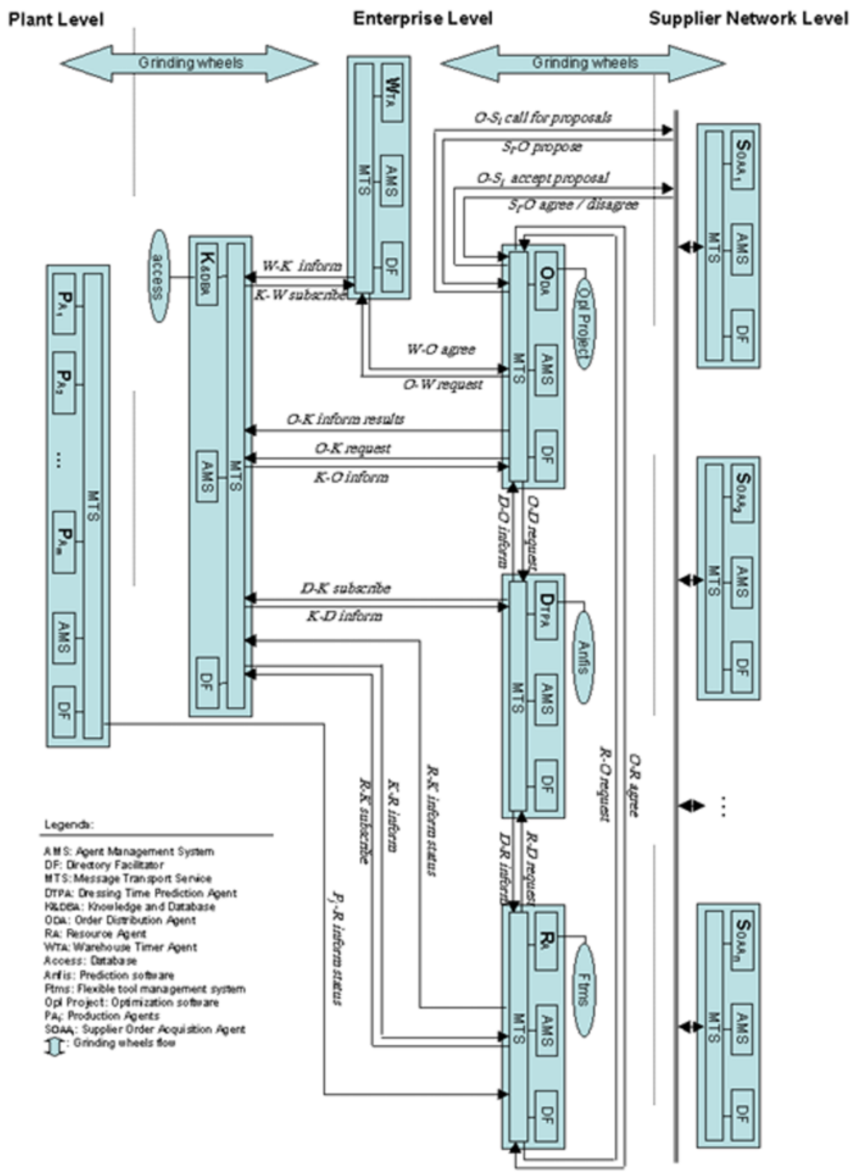


Fig. 10.1 Block schema of MATMS

- The RA, which merges the functions of tool inventory management, tool resource demand estimation and determination of tool order quantities; the RA domain-specific problem-solving function is the FTMS paradigm, dealt with in detail in Sect. 10.3. Initially, the RA receives information on CBN grinding wheel end-of-life events from the production agents (PA_j). For each CBN grinding wheel R/N, the RA requests a supplier-independent dressing cycle time prediction to the dressing time prediction agent (DTPA) (Teti and D'Addona, 2003b). Using this input and the information on production plans running at the plant level, it estimates the demand for CBN grinding wheel dressings. If it deems necessary to issue a dressing job order for a given CBN grinding wheel part-number, it requests the ODA to select an external tool manufacturer in the supply network for dressing job order allocation.
- The ODA, which selects the external supplier to which the CBN grinding wheel dressing job order should be allocated on the basis of negotiations and constraints. The ODA domain-specific problem-solving function for order allocation is implemented under ILOG OPL Studio 3.5 (Teti and D'Addona, 2003ac, Van Hentenryck, 1999).
- The DTPA, which carries out the predictions of CBN grinding wheel dressing cycle times, founded on historical data, on both a supplier-independent (Teti and D'Addona, 2003b) and a supplier-dependent basis (Teti and D'Addona, 2003c). The DTPA domain-specific problem-solving function is a neuro-fuzzy paradigm known as adaptive neuro-fuzzy inference system (Jang, 1993, 1995, Teti and D'Addona, 2004, Teti *et al.*, 2004).
- The knowledge and data base agent, which handles all the information relevant for tool management activities, including the updating of historical data on CBN grinding wheel dressing cycles.
- The WTA, which takes care of incoming and outgoing CBN grinding wheels, including the evaluation of actual dressing cycle times.

The plant level is responsible for the manufacturing of turbine blades. It comprises only one type of agent, the PA_j, representing the various production lines of the turbine blade producer factory. When a tool end-of-life event occurs, the relevant PA_j sends the worn-out CBN grinding wheel to the WTA and the corresponding information (P/N and S/N) to the RA to initiate the FTMS procedure. After dressing job completion, the external suppliers send the dressed CBN grinding wheels to the WTA, which forwards them to the appropriate PA.

10.3 Flexible Tool Management Strategies

The development and implementation of diverse FTMS is presented as an alternative to the traditional tool management procedure. Each FTMS is integrated in a MATMS for automatic tool procurement, designed to operate in the frame of a negotiation-based multiple-supplier network. The task of the FTMS is the optimal tool inventory sizing and control, including tool supply cost and stock-out risk minimization.

The main responsibility of the RA in the MATMS is the optimum tool inventory sizing and control of CBN grinding wheels for turbine blade fabrication, with reference to the minimization of tool management cost and stock-out risk. This task is carried out by the RA through its domain-specific problem-solving function, the FTMS paradigm.

Each time a worn-out CBN grinding wheel leaves a production line, a new P/N demand is set up as a function of the dressing cycle time. If an accurate estimate of the dressing cycle time of each worn-out CBN grinding wheel is provided by the DTPA, the demand for that P/N can be compared with the P/N on-hand inventory. Moreover, as a CBN grinding wheel purchase order issued at a certain time requires a suitable lead time (T_{pur}) before an actual delivery takes place, to avoid stock-out it is necessary to have available at any time a minimum stock at least equal to the P/N demand during the purchase order lead time T_{pur}.

An FTMS was initially presented in Teti and D'Addona (2003a, b) whose rationale was to make sure that the P/N on-hand inventory size, I, remains within an interval defined by two real-time control limits: the P/N demand during T_{pur} (lower limit, I_{min}) and the P/N demand calculated using the dressing cycle time predicted by the DTPA (upper limit, I_{max}). The inventory level, I, is left free to fluctuate within the limits $[I_{\mathrm{min}}, I_{\mathrm{max}}]$, provided neither of them is crossed, and whenever I decreases due to a tool wear-out event, the CBN grinding wheels are sent out for dressing. Otherwise, the turbine blade producer logistics must either provide additional S/Ns when the lower control limit is crossed ($I < I_{\mathrm{min}}$) or reduce the P/N on-hand inventory by suspending the dressing job order allocation when the upper control limit is crossed ($I > I_{\mathrm{max}}$) to bring back the stock level within the control range.

This FTMS paradigm was called planned demand rate approach as the P/N demand rate P/G is kept constant during the tool management period. The control range lower limit is constant and given by $I_{\mathrm{min}} = P/G \times T_{\mathrm{pur}}$, where T_{pur} is a constant as its historical records are only slightly variable. The control range upper limit, I_{max}, varies each time a tool wear-out event occurs as a function of the dressing time predictions, $T(j+1)$, provided by the DTPA:

$$I_{\mathrm{max}}(j) = P/G \times [T(j+1) + T_{\mathrm{int}}]$$

where j is the counter of worn-out wheels and T_{int} is the time to transfer the tool to its production line.

10.3.1 Current Inventory-Level Decision Making (INVADAPT)

The planned tool demand rate P/G is the ratio of two constant parameters, representing the average tool demand in the reference period. Thus, this parameter is unable to take into account any deviation from currently established production plans and/or current estimates of mean tool life. In fact, actual tool demand rates can be higher than the planned ones due to higher production volumes or unexpectedly shorter tool lives. They can also become lower than the planned ones because of lower production volumes. As a consequence, actual tool demand rates occurring in reality can notably differ from the planned tool demand rates and cause serious difficulties to the tool management activity.

To solve this problem, the concept of adaptive tool demand rate, $d_R(j)$, is introduced and defined as (Teti and D'Addona, 2003b):

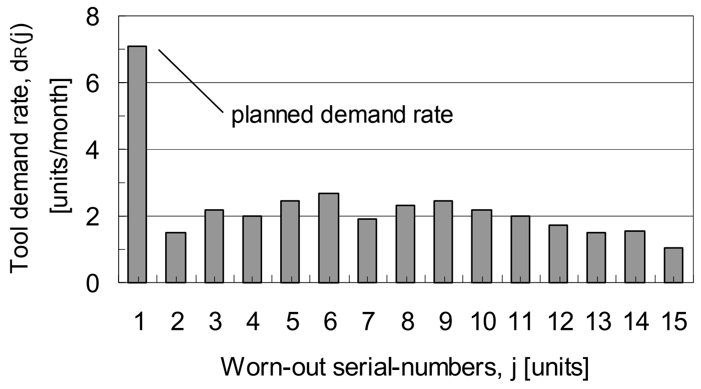
$$d_R(j) = \begin{cases} \dfrac{P}{G} & \text{if } t(j) \leq 1, \\ \dfrac{j}{t(j)} & \text{otherwise,} \end{cases} \tag{10.2}$$

where $t(j)$ is the time (in months), computed from the start of the FTMS procedure, at which the dressing operation for worn-out S/N is proposed. From Eq. 10.2, it can be noted that the planned tool demand rate P/G is taken as initial value for the adaptive tool demand rate $d_R(j)$. This value is kept constant during the first month of flexible tool management. In Figure 10.2, the adaptive tool demand rate for the three CBN grinding wheel P/Ns (M8110855, M8117351, M8137161) is reported *vs.* worn-out S/Ns j for a time period of one year. In the adaptive demand rate approach, the control limits are given by (Teti and D'Addona, 2003b):

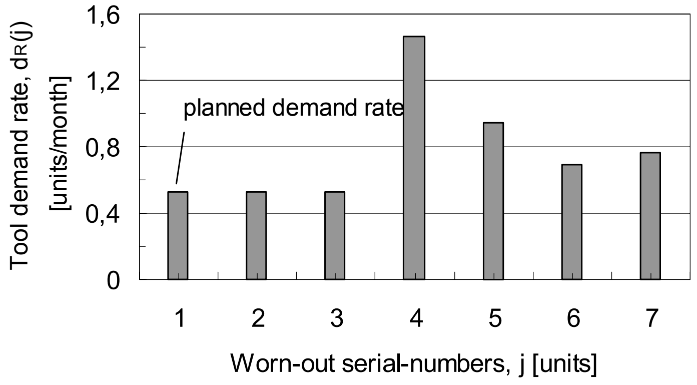
$$I_{\min}(j) = d_R(j) \times T_{\text{pur}} \text{ and } I_{\max}(j) = d_R(j) \times [\mathcal{T}(j+1) + T_{\text{int}}] \tag{10.3}$$

where j = counter of worn-out S/Ns, $\mathcal{T}(j+1)$ = DTPA dressing time prediction, T_{int} = internal time, i.e. time for a new or dressed grinding wheel to be transferred to the production line (constant for a given manufacturing plant). In Figure 10.3, the on-hand inventory control of a generic CBN grinding wheel P/N is described to illustrate the FTMS functioning for the INVADAPT approach.

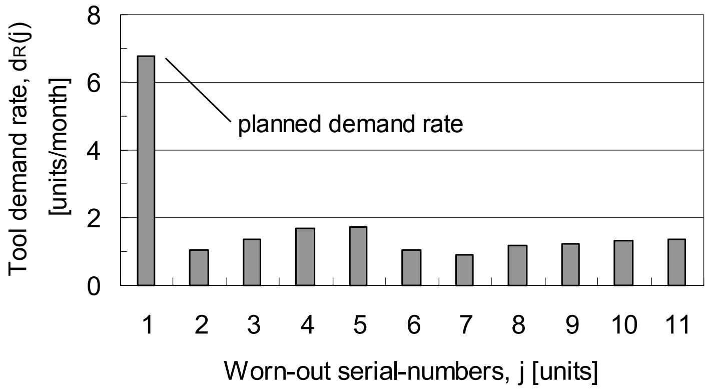
According to Eq. 10.3, as T_{int} is a constant typical of the manufacturing plant, T_{pur} is constant on a historical basis, and P/G is updated yearly by the turbine blade producer for each one-year period, the upper control limit $I_{\max}$ varies or stays constant as a function of the current adaptive demand rate $d_R(j)$ and the predicted dressing cycle time $\mathcal{T}(j+1)$, whereas the lower control limit $I_{\min}$ varies or stays constant only as a function of the current value of $d_R(j)$.



a



b



c

Fig. 10.2 Adaptive tool demand rate, $d_R(j)$, *vs*. worn-out S/Ns, *j*, in a period of one year, for P/Ns: **a** M8110855, **b** M8117351, **c** M8137161

At time $t = 0$, the P/N on-hand inventory size is I_0. Each time CBN grinding wheel wear-out (w-events in Figure 10.3), the P/N on-hand inventory level, I, decreases, the adaptive demand rate values are updated, and supplier-independent dressing cycle time predictions, $\mathcal{T}(j+1)$, are issued by the DTPA. Each time new or dressed CBN grinding wheels are delivered by the suppliers, the P/N on-hand inventory size, I, grows but the control limits I_{min} and I_{max} are not affected. However, if I crosses the upper control limit ($I > I_{\text{max}}$ at d_1 in Figure 10.3), at the next w-event (w_2 in Figure 10.3) $I - I_{\text{max}}$ worn-out S/Ns are kept on-hold in the enterprise warehouse (h_1 upper control limit crossing, Figure 10.3). After control limits updating, if I is within the control range the worn-out tools must be sent out for dressing but no tool purchase is required (w_1, w_3 in Figure 10.3). If I crosses the lower control limit ($I < I_{\text{min}}$ at w_4 in Figure 10.3), the worn-out tools must be sent out for dressing and further ($I_{\text{min}} - I$) S/Ns must be provided. If worn-out S/Ns are on-hold, they are sent out for dressing in partial or total substitution for new tool purchases (h_2 lower control limit crossing, Figure 10.3). At any rate, the number of required CBN grinding wheels exceeding the available on-hold worn-out S/Ns must be newly purchased.

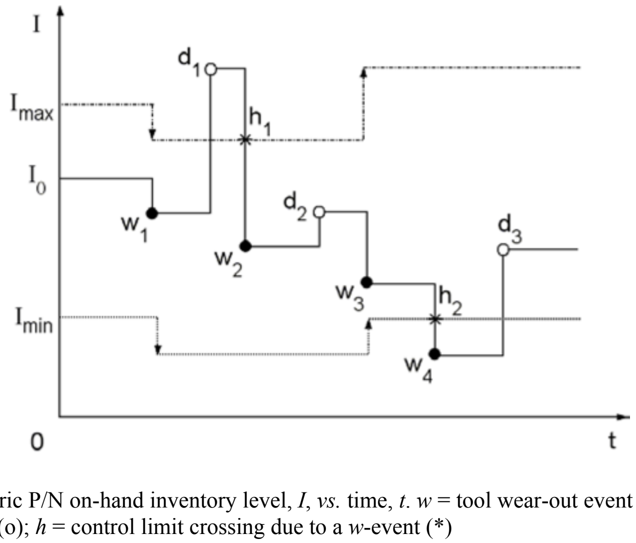


Fig. 10.3 Generic P/N on-hand inventory level, I, *vs.* time, t. w = tool wear-out event (•); d = tool delivery event (o); h = control limit crossing due to a w-event (*)

10.3.2 Fixed-Horizon Inventory-Level Decision Making (I-FUTURE)

The above version of the FTMS has the drawback of a “shortsighted” approach because decisions are taken only on the basis of the current inventory level, $I(t)$, with no consideration for what will happen in the future. To overcome this problem, a "non-shortsighted" version of the FTMS procedure, called I-FUTURE, is proposed whereby the FTMS takes into account for tool management decision-making the value of a future on-hand inventory level, $I_{\text{fut}}(t + T_{\text{rt}})$, instead of the current on-hand inventory level, $I(t)$. I_{fut} is the on-hand inventory level calculated with reference to a future time $(t + T_{\text{rt}})$ and is:

$$I_{\text{fut}}(t + T_{\text{rt}}) = I(t) + X + Y + Z - d_{\text{R}}(w) \times T_{\text{rt}} \tag{10.4}$$

where:

1. T_{rt} = reaction time of the FTMS.
2. X = number of worn-out tools sent out for dressing before the start of the FTMS procedure $(t < 0)$ and expected to be available at time $(t + T_{\text{rt}})$.
3. Y = number of worn-out tools sent out for dressing during the FTMS procedure $(t > 0)$ and predicted be available at time $(t + T_{\text{rt}})$.
4. Z = number of newly purchased worn-out tools not yet delivered: these will be positively available at time $(t + T_{\text{rt}})$ because purchase time $T_{\text{pur}} \leq T_{\text{rt}}$.
5. $d_{\text{R}}(w) \times T_{\text{rt}}$ = number of worn-out tools that will wearout during reaction time T_{rt} under constant d_{R} conditions.

In the first "non-shortsighted" approach I-FUTURE, if $I(t) \geq I_{\text{min}}$, T_{rt} is the sum of the mean historical dressing cycle times d_{ct} (6 weeks) and the internal time T_{int} (5 weeks):

$$T_{\text{rt}} = d_{\text{ct}} + T_{\text{int}} = 6 \text{ weeks} + 5 \text{ weeks} = 11 \text{ weeks}. \tag{10.5}$$

If $I(\text{t}) < I_{\text{min}}$, T_{rt} is considered equal to the purchase time T_{pur} (9 weeks):

$$T_{\text{rt}} = T_{\text{pur}} = 9 \text{ weeks}. \tag{10.6}$$

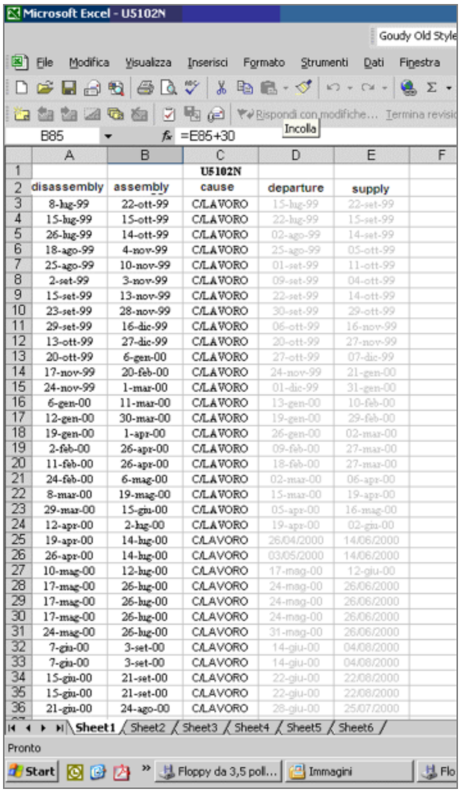
10.3.3 Variable-Horizon Inventory-Level Decision Making (INVADAPT_NS)

INVADAPT_NS is the second version of the "non-shortsighted" FTMS approach. In INVADAPT_NS, the FTMS reaction time T_{rt} is not considered equal to a constant value (9 or 11 weeks), like in I-FUTURE, but is given by the dressing cycle time prediction $\mathcal{T}(j+1)$ provided by the DTPA plus the value of the internal time T_{int}:

$$T_{\text{rt}} = \mathcal{T}(j+1) + T_{\text{int}}. \tag{10.7}$$

10.4 Tool Inventory Management Simulations

To carry out tool inventory management simulations through the FTMS paradigms, two files ".dat" must be created starting from the Excel® sheets received from the turbine blade producer (Figure 10.4).


Microsoft Excel - U5102N
Goudy Old Style
File Modifica Visualizza Inserisci Formato Strumenti Dati Finestra
Rispondi con modifiche... Termina revisi
Incolla
B85
=E85+30
Sheet1 Sheet2 Sheet3 Sheet4 Sheet5 Sheet6
Pronto
Start Floppy da 3,5 poll... Immagini Flo

	A	B	C	D	E	F
1			U5102N			
2	disassembly	assembly	cause	departure	supply	
3	8-lug-99	22-ott-99	C/LAVORO	15-lug-99	22-set-99	
4	15-lug-99	15-ott-99	C/LAVORO	22-lug-99	15-set-99	
5	26-lug-99	14-ott-99	C/LAVORO	02-ago-99	14-set-99	
6	18-ago-99	4-nov-99	C/LAVORO	25-ago-99	05-ott-99	
7	25-ago-99	10-nov-99	C/LAVORO	01-set-99	11-ott-99	
8	2-set-99	3-nov-99	C/LAVORO	09-set-99	04-ott-99	
9	15-set-99	13-nov-99	C/LAVORO	22-set-99	14-ott-99	
10	23-set-99	28-nov-99	C/LAVORO	30-set-99	29-ott-99	
11	29-set-99	16-dic-99	C/LAVORO	06-ott-99	16-nov-99	
12	13-ott-99	27-dic-99	C/LAVORO	20-ott-99	27-nov-99	
13	20-ott-99	6-gen-00	C/LAVORO	27-ott-99	07-dic-99	
14	17-nov-99	20-feb-00	C/LAVORO	24-nov-99	21-gen-00	
15	24-nov-99	1-mar-00	C/LAVORO	01-dic-99	31-gen-00	
16	6-gen-00	11-mar-00	C/LAVORO	13-gen-00	10-feb-00	
17	12-gen-00	30-mar-00	C/LAVORO	19-gen-00	29-feb-00	
18	19-gen-00	1-apr-00	C/LAVORO	26-gen-00	02-mar-00	
19	2-feb-00	26-apr-00	C/LAVORO	09-feb-00	27-mar-00	
20	11-feb-00	26-apr-00	C/LAVORO	18-feb-00	27-mar-00	
21	24-feb-00	6-mag-00	C/LAVORO	02-mar-00	06-apr-00	
22	8-mar-00	19-mag-00	C/LAVORO	15-mar-00	19-apr-00	
23	29-mar-00	15-giu-00	C/LAVORO	05-apr-00	16-mag-00	
24	12-apr-00	2-lug-00	C/LAVORO	19-apr-00	02-giu-00	
25	19-apr-00	14-lug-00	C/LAVORO	26/04/2000	14/06/2000	
26	26-apr-00	14-lug-00	C/LAVORO	03/05/2000	14/06/2000	
27	10-mag-00	12-lug-00	C/LAVORO	17-mag-00	12-giu-00	
28	17-mag-00	26-lug-00	C/LAVORO	24-mag-00	26/06/2000	
29	17-mag-00	26-lug-00	C/LAVORO	24-mag-00	26/06/2000	
30	17-mag-00	26-lug-00	C/LAVORO	24-mag-00	26/06/2000	
31	24-mag-00	26-lug-00	C/LAVORO	31-mag-00	26/06/2000	
32	7-giu-00	3-set-00	C/LAVORO	14-giu-00	04/08/2000	
33	7-giu-00	3-set-00	C/LAVORO	14-giu-00	04/08/2000	
34	15-giu-00	21-set-00	C/LAVORO	22-giu-00	22/08/2000	
35	15-giu-00	21-set-00	C/LAVORO	22-giu-00	22/08/2000	
36	21-giu-00	24-ago-00	C/LAVORO	28-giu-00	25/07/2000	

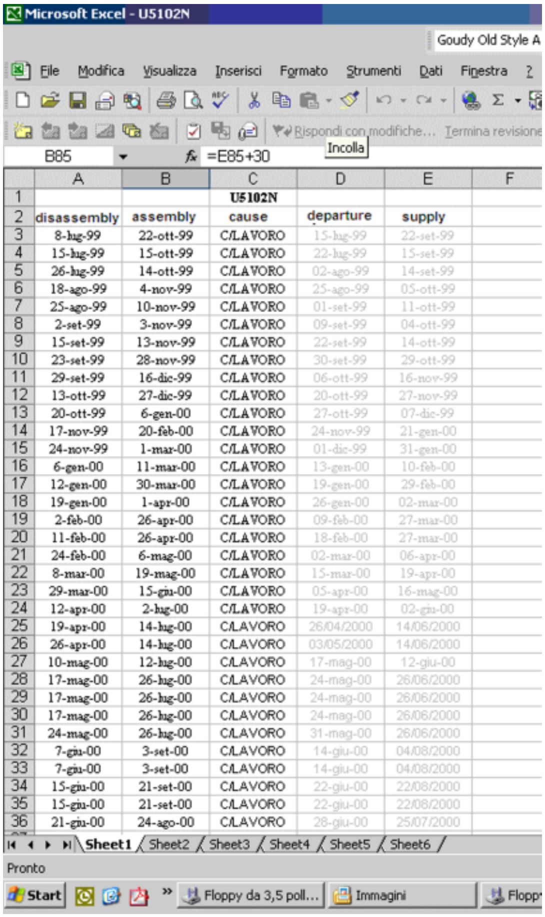
Fig. 10.4 Excel file (sheet 1) received by the turbine blade producer (P/N U5102N)

The original files are made of six sheets; for data file preparation, only sheets 1 and 2 are used. The information contained in sheet 1 is:

- DISASSEMBLY: date at which the worn-out CBN grinding wheel is disassembled.
- ASSEMBLY: date at which the dressed CBN grinding wheel is assembled for service.
- CAUSE: if the grinding wheel has been purchased (C/Construction), if it is worn-out (C/Job), if it has been sent for dressing but the dressing has been considered unnecessary (C/Check).

- DEPARTURE: dressing start date (it does not coincide with the DISASSEMBLY date since the worn-out CBN grinding wheels are sent to the suppliers in groups).
- SUPPLY: date at which the dressing is completed; it coincides with the date of arrival. To this initial information, two other items are added in two additional columns:
- SERIAL NUMBER: progressive number that identifies the single worn-out CBN grinding wheel.
- DRESSING TIME: time, in weeks, necessary for dressing [(ARRIVAL – DEPARTURE) / 7], only for C/Job and C/Check causes.

The file obtained is shown as an example in Figure 10.5.



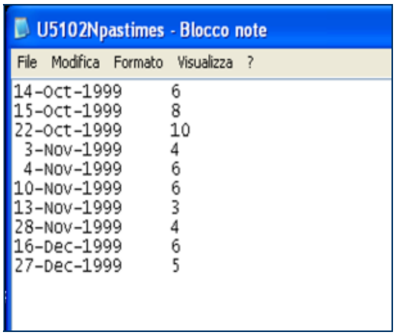
Microsoft Excel - U5102N
Goudy Old Style A
File Modifica Visualizza Inserisci Formato Strumenti Dati Finestra ?
Rispondi con modifiche... Termina revisione
Incolla
B85
=E85+30
Sheet1 Sheet2 Sheet3 Sheet4 Sheet5 Sheet6
Pronto
Start
Floppy da 3,5 poll...
Immagini

	A	B	C	D	E	F
1			U5102N			
2	disassembly	assembly	cause	departure	supply	
3	8-lug-99	22-ott-99	C/LAVORO	15-lug-99	22-set-99	
4	15-lug-99	15-ott-99	C/LAVORO	22-lug-99	15-set-99	
5	26-lug-99	14-ott-99	C/LAVORO	02-ago-99	14-set-99	
6	18-ago-99	4-nov-99	C/LAVORO	25-ago-99	05-ott-99	
7	25-ago-99	10-nov-99	C/LAVORO	01-set-99	11-ott-99	
8	2-set-99	3-nov-99	C/LAVORO	09-set-99	04-ott-99	
9	15-set-99	13-nov-99	C/LAVORO	22-set-99	14-ott-99	
10	23-set-99	28-nov-99	C/LAVORO	30-set-99	29-ott-99	
11	29-set-99	16-dic-99	C/LAVORO	06-ott-99	16-nov-99	
12	13-ott-99	27-dic-99	C/LAVORO	20-ott-99	27-nov-99	
13	20-ott-99	6-gen-00	C/LAVORO	27-ott-99	07-dic-99	
14	17-nov-99	20-feb-00	C/LAVORO	24-nov-99	21-gen-00	
15	24-nov-99	1-mar-00	C/LAVORO	01-dic-99	31-gen-00	
16	6-gen-00	11-mar-00	C/LAVORO	13-gen-00	10-feb-00	
17	12-gen-00	30-mar-00	C/LAVORO	19-gen-00	29-feb-00	
18	19-gen-00	1-apr-00	C/LAVORO	26-gen-00	02-mar-00	
19	2-feb-00	26-apr-00	C/LAVORO	09-feb-00	27-mar-00	
20	11-feb-00	26-apr-00	C/LAVORO	18-feb-00	27-mar-00	
21	24-feb-00	6-mag-00	C/LAVORO	02-mar-00	06-apr-00	
22	8-mar-00	19-mag-00	C/LAVORO	15-mar-00	19-apr-00	
23	29-mar-00	15-giu-00	C/LAVORO	05-apr-00	16-mag-00	
24	12-apr-00	2-lug-00	C/LAVORO	19-apr-00	02-giu-00	
25	19-apr-00	14-lug-00	C/LAVORO	26/04/2000	14/06/2000	
26	26-apr-00	14-lug-00	C/LAVORO	03/05/2000	14/06/2000	
27	10-mag-00	12-lug-00	C/LAVORO	17-mag-00	12-giu-00	
28	17-mag-00	26-lug-00	C/LAVORO	24-mag-00	26/06/2000	
29	17-mag-00	26-lug-00	C/LAVORO	24-mag-00	26/06/2000	
30	17-mag-00	26-lug-00	C/LAVORO	24-mag-00	26/06/2000	
31	24-mag-00	26-lug-00	C/LAVORO	31-mag-00	26/06/2000	
32	7-giu-00	3-set-00	C/LAVORO	14-giu-00	04/08/2000	
33	7-giu-00	3-set-00	C/LAVORO	14-giu-00	04/08/2000	
34	15-giu-00	21-set-00	C/LAVORO	22-giu-00	22/08/2000	
35	15-giu-00	21-set-00	C/LAVORO	22-giu-00	22/08/2000	
36	21-giu-00	24-ago-00	C/LAVORO	28-giu-00	25/07/2000	

Fig. 10.5 Obtained Excel file sheet 1 after the addition of S/N and dressing time (P/N U5102N)

For instance, in row 3 of Figure 10.5 "1" was added in column F to indicate the first grinding wheel in the simulation; and "10" was added in column G, equal to (E3 – D3) / 7, which must be rounded down or up to the nearest integer.

Using these data, the first file ".dat" can be created with the block notes. The name of this file comprises the P/N followed by the word "pastimes" and the extension ".dat" (this syntax must be followed rigidly otherwise the software cannot read the file). In the example, the file name is "U5102Npastimes.dat". The creation of the file starts by inserting in the block notes file a first column containing the last 10 dates of assembly of the year 1999 taken from column B of Figure 10.5 (in the example, from row 3 to row 12). If the number of dates of assembly for 1999 is less than 10, it is enough to take all the available ones. To this first column, a second column is added with the corresponding dressing times (column G)[2]. The obtained file is reported in Figure 10.6. If there are no dates of assembly for 1999, the dates that precede the year of simulation are used (to be verified).

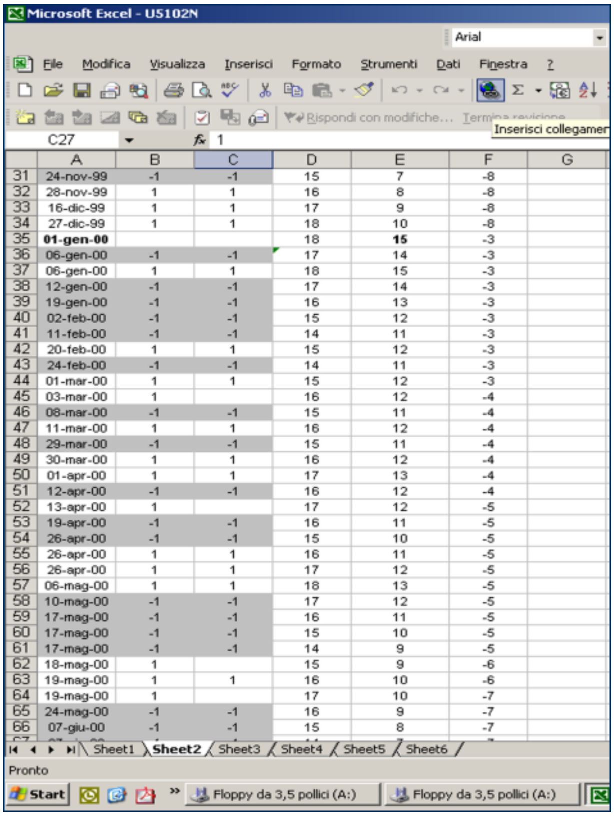

U5102Npastimes - Blocco note
File Modifica Formato Visualizza ?

```
14-Oct-1999     6
15-Oct-1999     8
22-Oct-1999     10
 3-Nov-1999     4
 4-Nov-1999     6
10-Nov-1999     6
13-Nov-1999     3
28-Nov-1999     4
16-Dec-1999     6
27-Dec-1999     5
```

Fig. 10.6 Block notes file for P/N U5102N pastimes data (first ".dat" file)

The name of the second file ".dat" is: P/N followed by the acronym "Ih" (inventory historical) and the extension ".dat". In the example, the name is "U5102NIh.dat". The information for the creation of this file is available in Excel file sheet 2 in Figure 10.7. In the page of the block notes file, a first column containing the dates of the events from 1 of January 2000 (column A, Figure 10.7) and a second column containing the corresponding availabilities (column D, Figure 10.7) are pasted. The obtained file is shown in Figure 10.8.

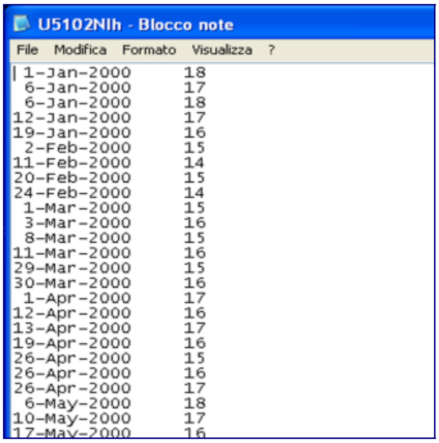
[2] To build this file, the proper information is copied from columns B and G and pasted into an Excel work sheet in two nearby columns and then from the work sheet to the block notes file.


Microsoft Excel - U5102N
Arial
File Modifica Visualizza Inserisci Formato Strumenti Dati Finestra ?
Rispondi con modifiche... Termina revisione
Inserisci collegamen
C27
Sheet1 Sheet2 Sheet3 Sheet4 Sheet5 Sheet6
Pronto
Start
Floppy da 3,5 pollici (A:)
Floppy da 3,5 pollici (A:)

`fx` 1

	A	B	C	D	E	F	G
31	24-nov-99	-1	-1	15	7	-8	
32	28-nov-99	1	1	16	8	-8	
33	16-dic-99	1	1	17	9	-8	
34	27-dic-99	1	1	18	10	-8	
35	**01-gen-00**			18	**15**	-3	
36	06-gen-00	-1	-1	17	14	-3	
37	06-gen-00	1	1	18	15	-3	
38	12-gen-00	-1	-1	17	14	-3	
39	19-gen-00	-1	-1	16	13	-3	
40	02-feb-00	-1	-1	15	12	-3	
41	11-feb-00	-1	-1	14	11	-3	
42	20-feb-00	1	1	15	12	-3	
43	24-feb-00	-1	-1	14	11	-3	
44	01-mar-00	1	1	15	12	-3	
45	03-mar-00	1		16	12	-4	
46	08-mar-00	-1	-1	15	11	-4	
47	11-mar-00	1	1	16	12	-4	
48	29-mar-00	-1	-1	15	11	-4	
49	30-mar-00	1	1	16	12	-4	
50	01-apr-00	1	1	17	13	-4	
51	12-apr-00	-1	-1	16	12	-4	
52	13-apr-00	1		17	12	-5	
53	19-apr-00	-1	-1	16	11	-5	
54	26-apr-00	-1	-1	15	10	-5	
55	26-apr-00	1	1	16	11	-5	
56	26-apr-00	1	1	17	12	-5	
57	06-mag-00	1	1	18	13	-5	
58	10-mag-00	-1	-1	17	12	-5	
59	17-mag-00	-1	-1	16	11	-5	
60	17-mag-00	-1	-1	15	10	-5	
61	17-mag-00	-1	-1	14	9	-5	
62	18-mag-00	1		15	9	-6	
63	19-mag-00	1	1	16	10	-6	
64	19-mag-00	1		17	10	-7	
65	24-mag-00	-1	-1	16	9	-7	
66	07-giu-00	-1	-1	15	8	-7	

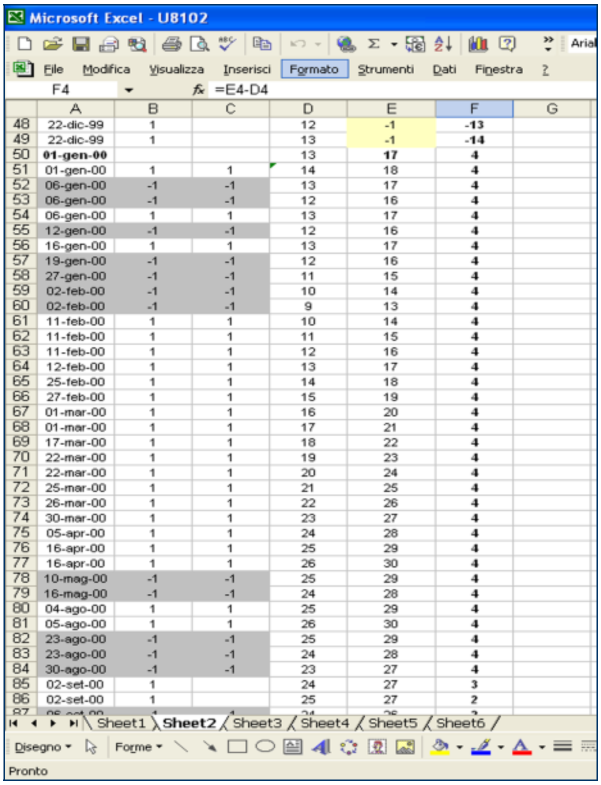
Fig. 10.7 Excel file sheet 2 for P/N U5102N. Column A: event date, B: real trend, C: ideal trend, D: availability, E: rig. x m.c., F: demand calculation



```
 1-Jan-2000      18
 6-Jan-2000      17
 6-Jan-2000      18
12-Jan-2000      17
19-Jan-2000      16
 2-Feb-2000      15
11-Feb-2000      14
20-Feb-2000      15
24-Feb-2000      14
 1-Mar-2000      15
 3-Mar-2000      16
 8-Mar-2000      15
11-Mar-2000      16
29-Mar-2000      15
30-Mar-2000      16
 1-Apr-2000      17
12-Apr-2000      16
13-Apr-2000      17
19-Apr-2000      16
26-Apr-2000      15
26-Apr-2000      16
26-Apr-2000      17
 6-May-2000      18
10-May-2000      17
17-May-2000      16
```

Fig. 10.8 Block notes file for P/N U5102NIh date (second ".dat" file)

For some files, it can happen that for the date of 1 January there are several rows (rows 50 and 51 of Figure 10.9). In this case, reference is made to column B, where it can be seen that B50 is empty while B51 contains "1"[3]. This means that on 1 January an event has occurred and the corresponding row to such event is now 51. So, in the file "Ih.dat" the date of 1 January is copied only once and the corresponding availability must be read in row 51. In the example of Figure 10.9, the availability at 1 January is therefore 14 and not 13.



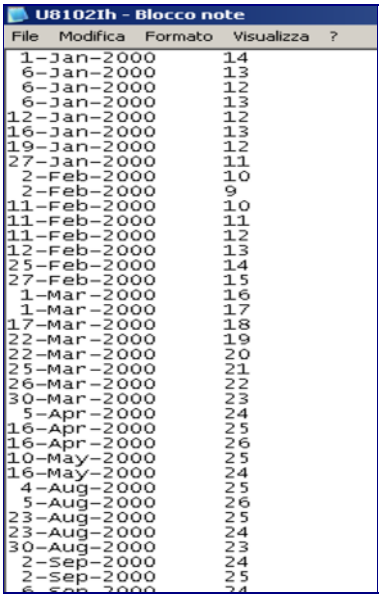
F4 =E4-D4

	A	B	C	D	E	F	G
48	22-dic-99	1		12	-1	-13	
49	22-dic-99	1		13	-1	-14	
50	**01-gen-00**			13	**17**	4	
51	01-gen-00	1	1	14	18	4	
52	06-gen-00	-1	-1	13	17	4	
53	06-gen-00	-1	-1	12	16	4	
54	06-gen-00	1	1	13	17	4	
55	12-gen-00	-1	-1	12	16	4	
56	16-gen-00	1	1	13	17	4	
57	19-gen-00	-1	-1	12	16	4	
58	27-gen-00	-1	-1	11	15	4	
59	02-feb-00	-1	-1	10	14	4	
60	02-feb-00	-1	-1	9	13	4	
61	11-feb-00	1	1	10	14	4	
62	11-feb-00	1	1	11	15	4	
63	11-feb-00	1	1	12	16	4	
64	12-feb-00	1	1	13	17	4	
65	25-feb-00	1	1	14	18	4	
66	27-feb-00	1	1	15	19	4	
67	01-mar-00	1	1	16	20	4	
68	01-mar-00	1	1	17	21	4	
69	17-mar-00	1	1	18	22	4	
70	22-mar-00	1	1	19	23	4	
71	22-mar-00	1	1	20	24	4	
72	25-mar-00	1	1	21	25	4	
73	26-mar-00	1	1	22	26	4	
74	30-mar-00	1	1	23	27	4	
75	05-apr-00	1	1	24	28	4	
76	16-apr-00	1	1	25	29	4	
77	16-apr-00	1	1	26	30	4	
78	10-mag-00	-1	-1	25	29	4	
79	16-mag-00	-1	-1	24	28	4	
80	04-ago-00	1	1	25	29	4	
81	05-ago-00	1	1	26	30	4	
82	23-ago-00	-1	-1	25	29	4	
83	23-ago-00	-1	-1	24	28	4	
84	30-ago-00	-1	-1	23	27	4	
85	02-set-00	1		24	27	3	
86	02-set-00	1		25	27	2	

Fig. 10.9 Excel file sheet 2 for P/N U8102

In Figure 10.10, the file "U8102Ih.dat" is shown. The built ".dat" files must be inputted in the Work directory of MatLab. The two ".dat" files have been built using a historical data base so to allow the FTMS to make reliable forecasts of the dressing times and to calculate the tool demand rate. These data, in fact, allow the software to decide whether a worn-out CBN grinding wheel should be sent for dressing or not.

[3] If column B contains "1" or "-1," an event has happened.



```
 1-Jan-2000      14
 6-Jan-2000      13
 6-Jan-2000      12
 6-Jan-2000      13
12-Jan-2000      12
16-Jan-2000      13
19-Jan-2000      12
27-Jan-2000      11
 2-Feb-2000      10
 2-Feb-2000      9
11-Feb-2000      10
11-Feb-2000      11
11-Feb-2000      12
12-Feb-2000      13
25-Feb-2000      14
27-Feb-2000      15
 1-Mar-2000      16
 1-Mar-2000      17
17-Mar-2000      18
22-Mar-2000      19
22-Mar-2000      20
25-Mar-2000      21
26-Mar-2000      22
30-Mar-2000      23
 5-Apr-2000      24
16-Apr-2000      25
16-Apr-2000      26
10-May-2000      25
16-May-2000      24
 4-Aug-2000      25
 5-Aug-2000      26
23-Aug-2000      25
23-Aug-2000      24
30-Aug-2000      23
 2-Sep-2000      24
 2-Sep-2000      25
 6-Sep-2000      24
```

Fig. 10.10 Block notes file for P/N U8102Ih data

In order to carry out a simulation, an Excel file must be created. This file is built starting from the available data in the file of Figure 10.5 (that is the original Excel file to which the column of the S/N and the column of the dressing time have been added). This is an Excel sheet (and not MatLab) from which we will read the information to input into the simulation; it is made up of five columns (Figure 10.11):

1. DATE: in the first column, the dates of disassembly and assembly events related to the year 2000 taken from column A (starting from row 16) and from column B (starting from row 13) of Figure 10.5 are listed together in chronological order.
2. TYPE: in the second column, the type of event is indicated:
   – s (for shipment) for a disassembly date and
   – d (for delivery) for an assembly date.
3. NUMBER: it counts the number of events for each date (in this file, in each row, only one event is listed).
4. SERIAL NUMBER: this is retrieved from column F of Figure 10.5.
5. DRESSING TIME: this is retrieved from column G of Figure 10.5.

The procedure for building the above file is the following:

1. Copy the dates of disassembly from column A of Figure 10.5 and paste them in the first column of a temporary Excel sheet, insert in the second column the letter "s", insert in the third column the number "1" and in the fourth column the corresponding S/Ns (they must be retrieved from column F of Figure 10.5).

2. Copy the dates of assembly from column B of Figure 10.5 and paste them in the first column of the aforesaid Excel sheet under the preceding dates, insert in the second column the letter "d", insert in the third column the number "1", in the fourth column the corresponding S/Ns and in the fifth column the corresponding dressing time (they must be retrieved from column G of Figure 10.5).
3. The resulting five columns must be ordered by selecting them and sorting them in increasing order in comparison with the first column. If, on the same date, a "shipment" event and a "delivery" event occur simultaneously, the increasing order is established using the second column.

10.4.1 Simulation Through INVADAPT

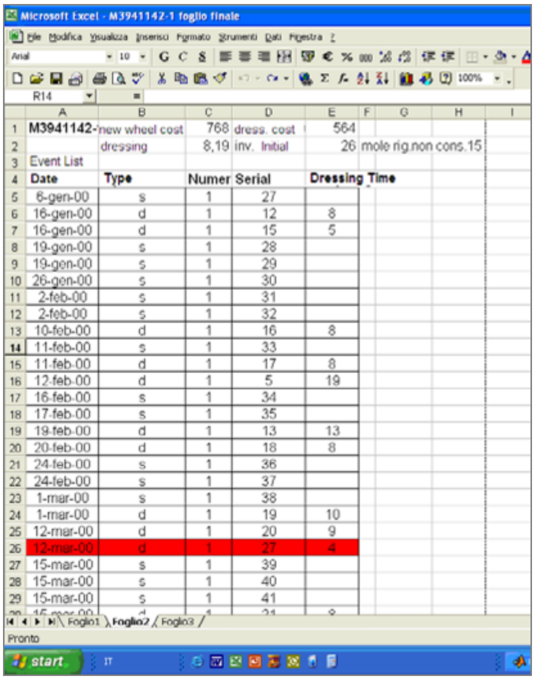
To begin the simulation, in the command window of MatLab, the word “INVADAPT” must be typed and the “enter” key pressed. In the command window all information related to the P/N to be simulated must be inputted. The first information to be input is the name of the P/N to simulate, then the date at which the simulation must start: 1 January 2000 (it must be input using the format 01-jan-2000).

Subsequently, the purchase cost of the CBN grinding wheel must be inputted. In the same way, the following information must be inputted: dressing cost, planned demand rate (given by the ratio between the planned number of pieces to produce *P* and the average life of the grinding wheel *G*), initial on-hand inventory (number of grinding wheels that are available at the start of the simulation), and number of grinding wheels that have been sent for dressing before the start of the simulation (the starting date inputted in the command window), and that have not been yet delivered. Between an information item and the following, the “enter” key must be pressed.

After having inputted all these information items "enter" is pressed once again. The request "event date" appears on screen where the information related to the first event must be inputted: the date of the event (using the format 01-jan-2000), the type of event ("shipment" if the grinding wheel is worn out, "delivery" if the dressed grinding wheel is delivered), and the number of grinding wheels related to such event.

By pressing the “enter” key, a number of rows equal to the number of grinding wheels for the event appear. In these rows the S/Ns must be inputted and, in case of delivery, an equal number of rows also appear where the corresponding dressing cycle times must be inputted. The system, on the basis of this information, makes the decision related to the management of the worn-out CBN grinding wheels.

An example of simulation and the operations that must be carried out to transform the historical list into the adaptive one[4] are reported in the following. A special sheet must be built by copying the "events list" of Figure 10.11 for convenience. All the changes to the events list during the simulation will be made in this sheet (in this way, the original information will not be lost).



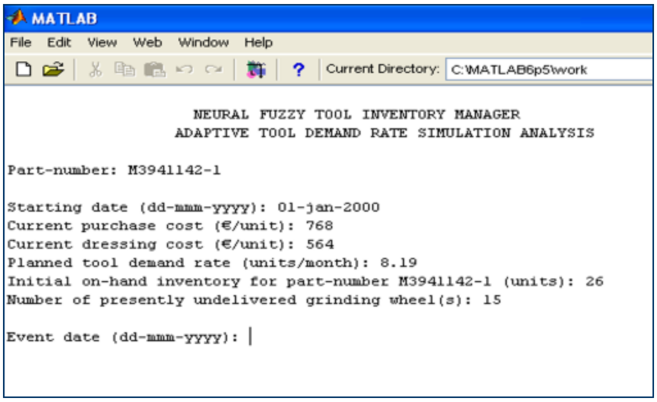
Microsoft Excel - M3941142-1 foglio finale

	A	B	C	D	E	F	G	H
1	M3941142-	new wheel cost	768	dress. cost	564			
2		dressing	8,19	inv. Initial	26	mole rig.non cons.15		
3	Event List							
4	Date	Type	Numer	Serial	Dressing Time			
5	6-gen-00	s	1	27				
6	16-gen-00	d	1	12	8			
7	16-gen-00	d	1	15	5			
8	19-gen-00	s	1	28				
9	19-gen-00	s	1	29				
10	26-gen-00	s	1	30				
11	2-feb-00	s	1	31				
12	2-feb-00	s	1	32				
13	10-feb-00	d	1	16	8			
14	11-feb-00	s	1	33				
15	11-feb-00	d	1	17	8			
16	12-feb-00	d	1	5	19			
17	16-feb-00	s	1	34				
18	17-feb-00	s	1	35				
19	19-feb-00	d	1	13	13			
20	20-feb-00	d	1	18	8			
21	24-feb-00	s	1	36				
22	24-feb-00	s	1	37				
23	1-mar-00	s	1	38				
24	1-mar-00	d	1	19	10			
25	12-mar-00	d	1	20	9			
26	12-mar-00	d	1	27	4			
27	15-mar-00	s	1	39				
28	15-mar-00	s	1	40				
29	15-mar-00	s	1	41				

Fig. 10.11 Black rows indicate that they must not be considered during simulation

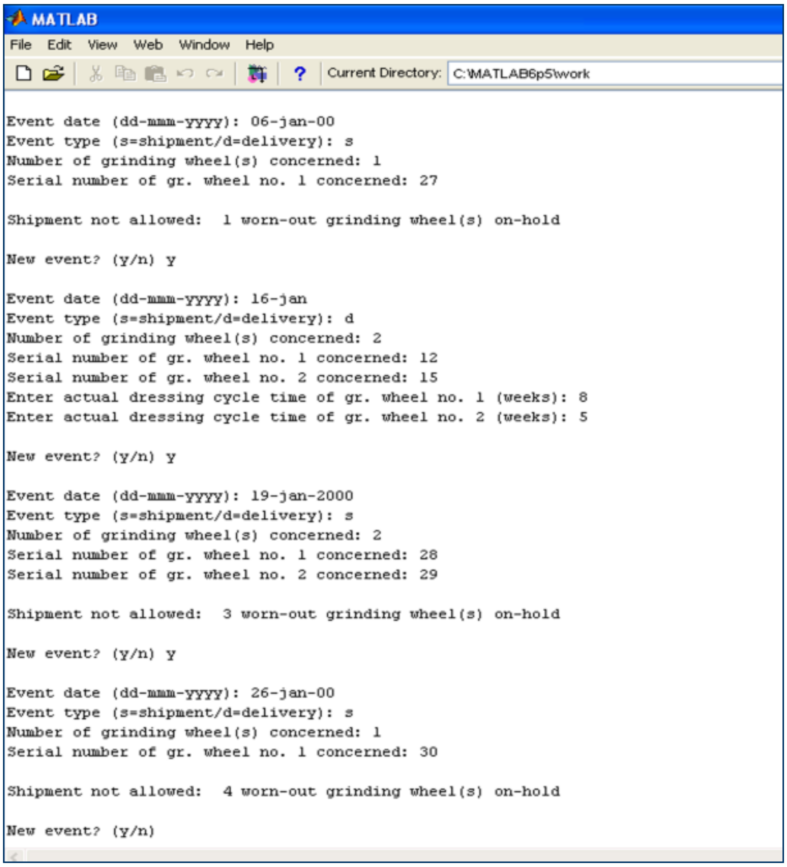
The simulation starts by inputting the initial data in the command window (Figure 10.12). After inserting the first event (grinding S/N number 27 of row 5 in Figure 10.11), the next event is inserted (grinding wheel S/N 12 of row 6 in Figure 10.11) as shown in Figure 10.13. By pressing the “enter” key, the software decides if the grinding wheel must be sent to an external supplier for dressing or kept “on hold” in the tool warehouse. If it is decided to keep the grinding wheel S/N 27 "on hold" in the warehouse, in the sheet of Figure 10.11 the row related to the delivery event (d in the type column) of grinding wheel S/N 27 (Figure 10.11, row 26 in black) must be cancelled. Likewise, by proceeding to the next event it can be observed that also grinding wheel S/Ns 28 and 29 are kept "on hold" (Figure 10.14).

[4] Adaptive list: the original file of Figure 10.11 modified according to the decisions of the FTMS.


MATLAB
File Edit View Web Window Help
Current Directory: C:\MATLAB6p5\work

```
                    NEURAL FUZZY TOOL INVENTORY MANAGER
                  ADAPTIVE TOOL DEMAND RATE SIMULATION ANALYSIS

Part-number: M3941142-1

Starting date (dd-mmm-yyyy): 01-jan-2000
Current purchase cost (€/unit): 768
Current dressing cost (€/unit): 564
Planned tool demand rate (units/month): 8.19
Initial on-hand inventory for part-number M3941142-1 (units): 26
Number of presently undelivered grinding wheel(s): 15

Event date (dd-mmm-yyyy): |
```

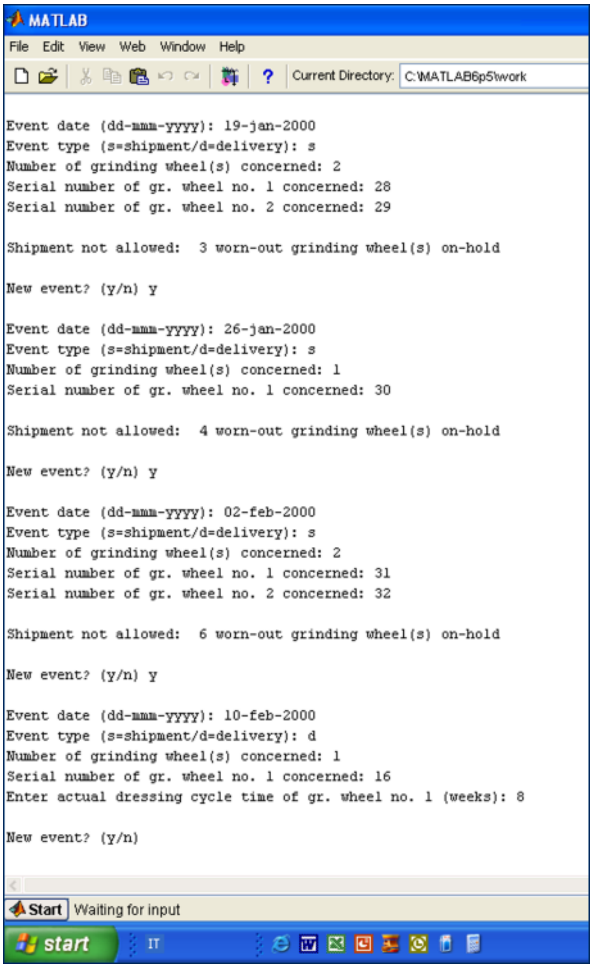
Fig. 10.12 Input of initial data into the command window


MATLAB
File Edit View Web Window Help
Current Directory: C:\MATLAB6p5\work

```
Event date (dd-mmm-yyyy): 06-jan-00
Event type (s=shipment/d=delivery): s
Number of grinding wheel(s) concerned: 1
Serial number of gr. wheel no. 1 concerned: 27

Shipment not allowed:  1 worn-out grinding wheel(s) on-hold

New event? (y/n) y

Event date (dd-mmm-yyyy): 16-jan
Event type (s=shipment/d=delivery): d
Number of grinding wheel(s) concerned: 2
Serial number of gr. wheel no. 1 concerned: 12
Serial number of gr. wheel no. 2 concerned: 15
Enter actual dressing cycle time of gr. wheel no. 1 (weeks): 8
Enter actual dressing cycle time of gr. wheel no. 2 (weeks): 5

New event? (y/n) y

Event date (dd-mmm-yyyy): 19-jan-2000
Event type (s=shipment/d=delivery): s
Number of grinding wheel(s) concerned: 2
Serial number of gr. wheel no. 1 concerned: 28
Serial number of gr. wheel no. 2 concerned: 29

Shipment not allowed:  3 worn-out grinding wheel(s) on-hold

New event? (y/n) y

Event date (dd-mmm-yyyy): 26-jan-00
Event type (s=shipment/d=delivery): s
Number of grinding wheel(s) concerned: 1
Serial number of gr. wheel no. 1 concerned: 30

Shipment not allowed:  4 worn-out grinding wheel(s) on-hold

New event? (y/n)
```

Fig. 10.13 Input of event information into the command window and response from the system for S/N 27

For the event dated 10 February 2000 related to worn-out grinding wheel S/N 16, the software decides to send it for dressing (pointing out the date at which it is predicted to be available again). In this case, the sheet of Figure 10.11 is not modified.

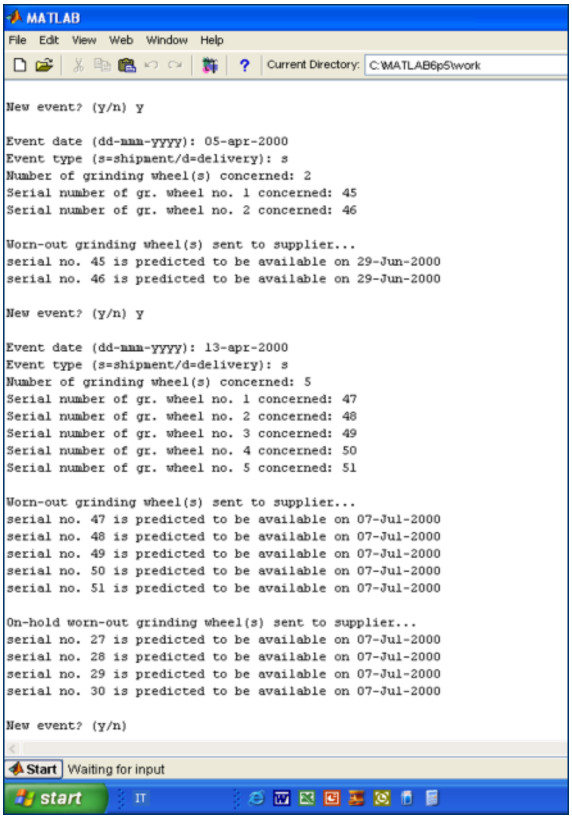

MATLAB
File Edit View Web Window Help
Current Directory: C:\MATLAB6p5\work
Start Waiting for input
start

```
Event date (dd-mmm-yyyy): 19-jan-2000
Event type (s=shipment/d=delivery): s
Number of grinding wheel(s) concerned: 2
Serial number of gr. wheel no. 1 concerned: 28
Serial number of gr. wheel no. 2 concerned: 29

Shipment not allowed:  3 worn-out grinding wheel(s) on-hold

New event? (y/n) y

Event date (dd-mmm-yyyy): 26-jan-2000
Event type (s=shipment/d=delivery): s
Number of grinding wheel(s) concerned: 1
Serial number of gr. wheel no. 1 concerned: 30

Shipment not allowed:  4 worn-out grinding wheel(s) on-hold

New event? (y/n) y

Event date (dd-mmm-yyyy): 02-feb-2000
Event type (s=shipment/d=delivery): s
Number of grinding wheel(s) concerned: 2
Serial number of gr. wheel no. 1 concerned: 31
Serial number of gr. wheel no. 2 concerned: 32

Shipment not allowed:  6 worn-out grinding wheel(s) on-hold

New event? (y/n) y

Event date (dd-mmm-yyyy): 10-feb-2000
Event type (s=shipment/d=delivery): d
Number of grinding wheel(s) concerned: 1
Serial number of gr. wheel no. 1 concerned: 16
Enter actual dressing cycle time of gr. wheel no. 1 (weeks): 8

New event? (y/n)
```

Fig. 10.14 Event info input into the command window and system response for S/Ns 28, 29, 30, 31, 32, 16

The event dated 13 April 2000 (related to grinding wheel S/Ns 47, 48, 49, 50, 51 of rows 44, 45, 46, 47, 48 in Figure 10.11) at which the shipment of grinding wheel S/Ns 47, 48, 49, 50, 51 occurs is inputted. The software takes the decisions shown in Figure 10.15. From Figure 10.15, it can be noted that INVADAPT decided to send for dressing not only the worn out grinding wheels 47, 48, 49, 50, 51, but also the grinding wheels that were kept "on hold", 27, 28, 29, 30, pointing out the date at which they are predicted to be available again.

Therefore, the date at which grinding wheels 27, 28, 29, 30 will return must be calculated: to the date of actual shipment (it does not coincide with the disassembly date) the corresponding dressing cycle time and the internal time must be added. The "internal time" is the time interval between the grinding wheel delivery from the supplier and the assembly date: in our case, this time is equal to 5 weeks and is considered constant.

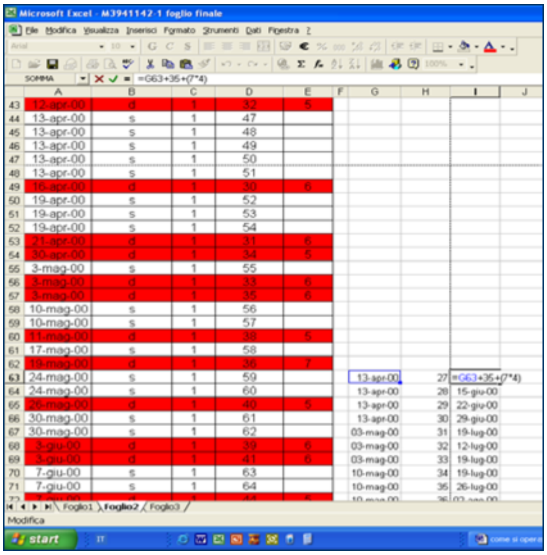

MATLAB
File Edit View Web Window Help
Current Directory: C:\MATLAB6p5\work
Start
Waiting for input
start
IT

```
New event? (y/n) y

Event date (dd-mmm-yyyy): 05-apr-2000
Event type (s=shipment/d=delivery): s
Number of grinding wheel(s) concerned: 2
Serial number of gr. wheel no. 1 concerned: 45
Serial number of gr. wheel no. 2 concerned: 46

Worn-out grinding wheel(s) sent to supplier...
serial no. 45 is predicted to be available on 29-Jun-2000
serial no. 46 is predicted to be available on 29-Jun-2000

New event? (y/n) y

Event date (dd-mmm-yyyy): 13-apr-2000
Event type (s=shipment/d=delivery): s
Number of grinding wheel(s) concerned: 5
Serial number of gr. wheel no. 1 concerned: 47
Serial number of gr. wheel no. 2 concerned: 48
Serial number of gr. wheel no. 3 concerned: 49
Serial number of gr. wheel no. 4 concerned: 50
Serial number of gr. wheel no. 5 concerned: 51

Worn-out grinding wheel(s) sent to supplier...
serial no. 47 is predicted to be available on 07-Jul-2000
serial no. 48 is predicted to be available on 07-Jul-2000
serial no. 49 is predicted to be available on 07-Jul-2000
serial no. 50 is predicted to be available on 07-Jul-2000
serial no. 51 is predicted to be available on 07-Jul-2000

On-hold worn-out grinding wheel(s) sent to supplier...
serial no. 27 is predicted to be available on 07-Jul-2000
serial no. 28 is predicted to be available on 07-Jul-2000
serial no. 29 is predicted to be available on 07-Jul-2000
serial no. 30 is predicted to be available on 07-Jul-2000

New event? (y/n)
```

Fig. 10.15 Event info input into the command window and system response for S/Ns 27, 28, 29, 30, 47, 48, 49, 50, 51

Then, the actual shipment dates for S/Ns 27, 28, 29, 30 do not coincide with the historical shipment dates of S/Ns 27, 28, 29, 30. The dates must be read in column A, rows 7–9 of the file in Figure 10.11: 6 January 2000 (27), 19 January 2000 (28, 29) 26 January 2000 (30).

By adding to the dates of actual shipment the dressing cycle time related to each S/N (retrieved from column E of the file in Figure 10.11) and the internal time equal to 5 weeks, the obtained dates are: 15 June 2000 (27, 28), 22 June 2000 (29) and 29 June 2000 (30) (Figure 10.16).



Microsoft Excel - M3941142-1 foglio finale

SOMMA =G63+35+(7*4)

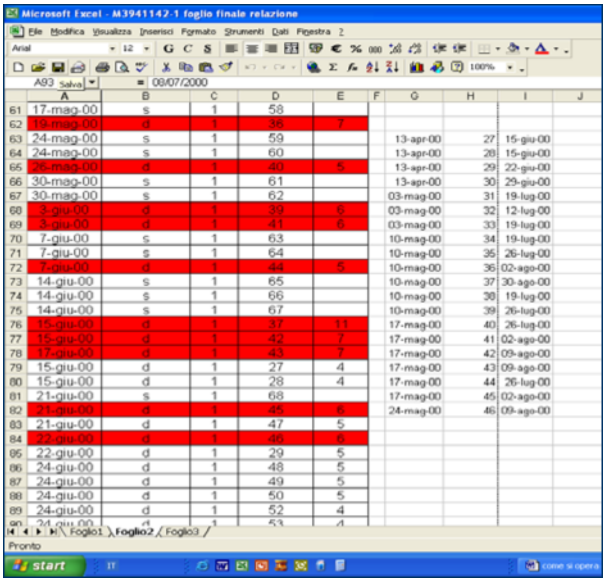
	A	B	C	D	E	F	G	H	I	J
43	12-apr-00	d	1	32	5					
44	13-apr-00	s	1	47						
45	13-apr-00	s	1	48						
46	13-apr-00	s	1	49						
47	13-apr-00	s	1	50						
48	13-apr-00	s	1	51						
49	16-apr-00	d	1	30	6					
50	19-apr-00	s	1	52						
51	19-apr-00	s	1	53						
52	19-apr-00	s	1	54						
53	21-apr-00	d	1	31	6					
54	30-apr-00	d	1	34	5					
55	3-mag-00	s	1	55						
56	3-mag-00	d	1	33	6					
57	3-mag-00	d	1	35	6					
58	10-mag-00	s	1	56						
59	10-mag-00	s	1	57						
60	11-mag-00	d	1	38	5					
61	17-mag-00	s	1	58						
62	19-mag-00	d	1	36	7					
63	24-mag-00	s	1	59			13-apr-00	27	=G63+35+(7*4)	
64	24-mag-00	s	1	60			13-apr-00	28	15-giu-00	
65	26-mag-00	d	1	40	5		13-apr-00	29	22-giu-00	
66	30-mag-00	s	1	61			13-apr-00	30	29-giu-00	
67	30-mag-00	s	1	62			03-mag-00	31	19-lug-00	
68	3-giu-00	d	1	39	6		03-mag-00	32	12-lug-00	
69	3-giu-00	d	1	41	6		03-mag-00	33	19-lug-00	
70	7-giu-00	s	1	63			10-mag-00	34	19-lug-00	
71	7-giu-00	s	1	64			10-mag-00	35	26-lug-00	
72	7-giu-00	d	1	44	5		10-mag-00	36	02-ago-00	

Foglio1 Foglio2 Foglio3

Modifica

Fig. 10.16 Actual return time calculation

In the adaptive list, four rows related to the delivery events of S/Ns 27, 28, 29, 30, are added by modifying the historical list of events that coincides with the updated sheet in Figure 10.17. The rows related to this last operation are rows 79, 80, 85, 86 of Figure10.17.



Microsoft Excel - M3941142-1 foglio finale relazione

A93 = 08/07/2000

	A	B	C	D	E	F	G	H	I	J
61	17-mag-00	s	1	58						
62	19-mag-00	d	1	36	7					
63	24-mag-00	s	1	59			13-apr-00	27	15-giu-00	
64	24-mag-00	s	1	60			13-apr-00	28	15-giu-00	
65	26-mag-00	d	1	40	5		13-apr-00	29	22-giu-00	
66	30-mag-00	s	1	61			13-apr-00	30	29-giu-00	
67	30-mag-00	s	1	62			03-mag-00	31	19-lug-00	
68	3-giu-00	d	1	39	6		03-mag-00	32	12-lug-00	
69	3-giu-00	d	1	41	6		03-mag-00	33	19-lug-00	
70	7-giu-00	s	1	63			10-mag-00	34	19-lug-00	
71	7-giu-00	s	1	64			10-mag-00	35	26-lug-00	
72	7-giu-00	d	1	44	5		10-mag-00	36	02-ago-00	
73	14-giu-00	s	1	65			10-mag-00	37	30-ago-00	
74	14-giu-00	s	1	66			10-mag-00	38	19-lug-00	
75	14-giu-00	s	1	67			10-mag-00	39	26-lug-00	
76	15-giu-00	d	1	37	11		17-mag-00	40	26-lug-00	
77	15-giu-00	d	1	42	7		17-mag-00	41	02-ago-00	
78	17-giu-00	d	1	43	7		17-mag-00	42	09-ago-00	
79	15-giu-00	d	1	27	4		17-mag-00	43	09-ago-00	
80	15-giu-00	d	1	28	4		17-mag-00	44	26-lug-00	
81	21-giu-00	s	1	68			17-mag-00	45	02-ago-00	
82	21-giu-00	d	1	45	6		24-mag-00	46	09-ago-00	
83	21-giu-00	d	1	47	5					
84	22-giu-00	d	1	46	6					
85	22-giu-00	d	1	29	5					
86	24-giu-00	d	1	48	5					
87	24-giu-00	d	1	49	5					
88	24-giu-00	d	1	50	5					
89	24-giu-00	d	1	52	4					
90	24-giu-00	d	1	53	4					

Foglio1 Foglio2 Foglio3

Pronto

Fig. 10.17 Sheet of Figure 10.11 after updating

All the other events are dealt with in this way, observing that INVADAPT, on 3 May 2000, decides to purchase a new grinding wheel (it also happens on 10 May 2000, 17 May 2000 and 24 May 2000). After the last event, at the end of the simulation, the P/N inventory level trend is graphically visualized as in Figure 10.18. This figure allows for the comparison of the simulated inventory level trend I of the INVADAPT management with the historical trend and also shows the two inventory level control range limits $I_{\min}$ and $I_{\max}$.

INVADAPT also visualizes the plot of the "tool supply cost", shown in Figure 10.19, and the plot of the "adaptive tool demand rate" values, shown in Figure 10.20.

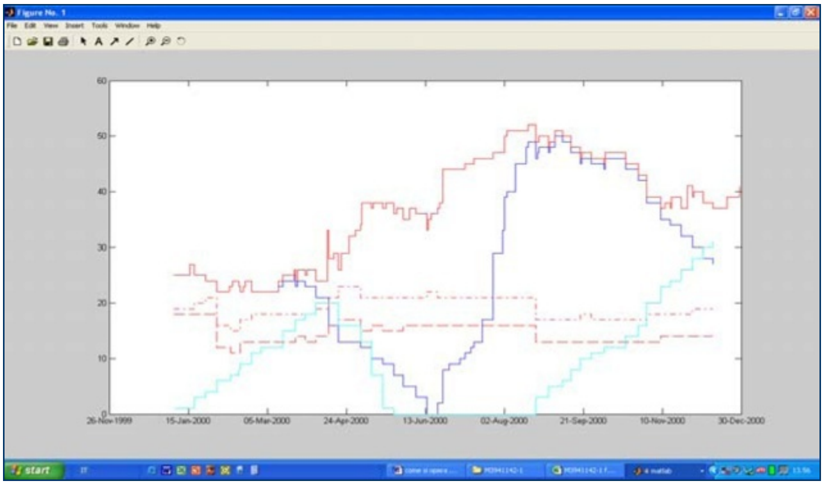


Fig. 10.18 CBN grinding-wheel inventory trend for P/N M3941142-1

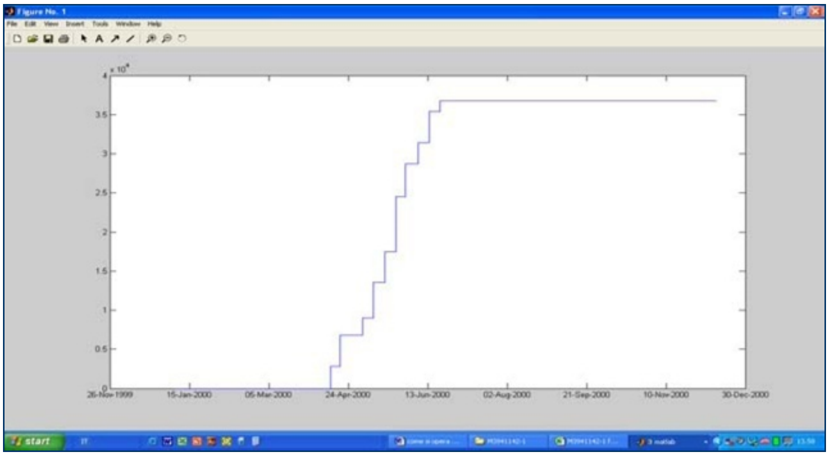


Fig. 10.19 Tool supply cost plot for P/N M3941142-1

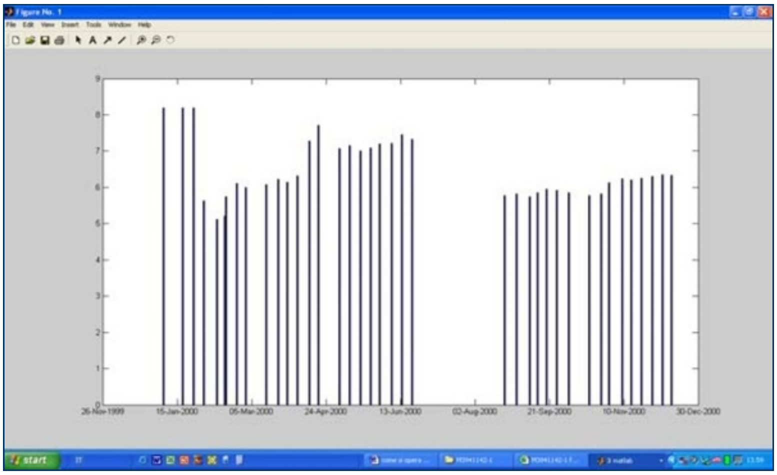


Fig. 10.20 Adaptive tool demand rate plot for P/N M3941142-1

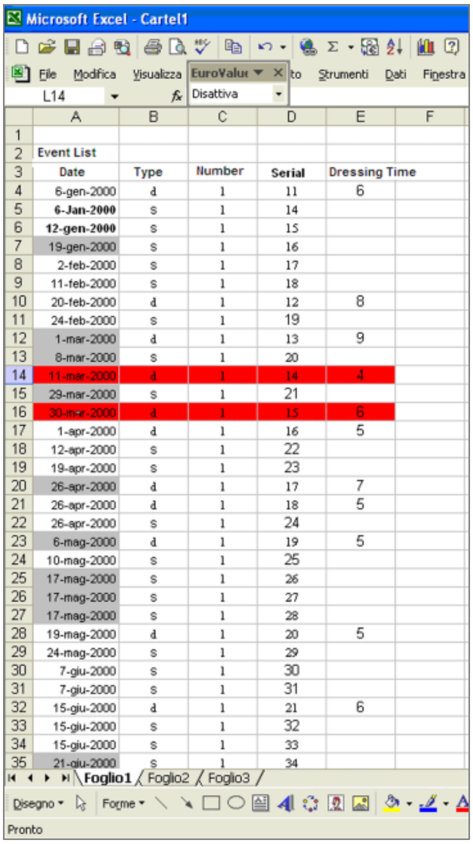
10.4.2 Simulation Through I-FUTURE

To begin the simulation, in the main screen of I-FUTURE, one clicks on the box "new simulation". The initialization window appears in which all the information related to the P/N to be simulated must be inputted. The first information item to be inputted is the start date of the simulation: 1 January 2000. Then, a combo box can be opened where the P/N to simulate is selected.

Subsequently, pressing the tab key, the text box appears in which the purchase cost of the CBN grinding wheel is inputted. In the same way, the following information must be inputted: dressing cost, planned demand rate (given by the ratio between the planned number of pieces to produce *P* and the average life of the grinding wheel *G*), initial on-hand inventory (number of grinding wheels that are available at the start of the simulation), and number of grinding wheels that have been sent for dressing before the start of the simulation (the starting date inputted in the command window), and that have not been yet delivered. After having inputted these information items one presses "OK". The window "event information" appears in which the information related to the first event is inputted: the date of the event (in the international format, for example 1-jan-2000), the type of event ("shipment" if the grinding wheel is worn out, "delivery" if the dressed grinding wheel is delivered), and the number of grinding wheels related to such event.

By pressing the tab key, a number of rows equal to the number of grinding wheels for the event appear. In these rows the S/Ns must be inputted and, in case of delivery, an equal number of rows also appear where the corresponding dressing cycle times must be inputted. The system, on the basis of this information, makes the decision related to the management of the worn-out CBN grinding wheels. An example of simulation and the operations that must be carried out to transform the historical list into the adaptive one are reported in the following.

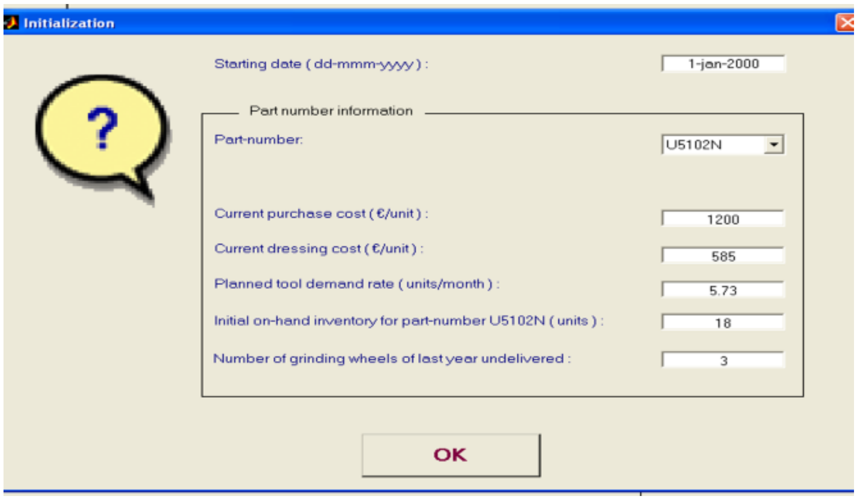
The best practice is to build a special sheet copying the "events list" of Figure 10.21. All the changes to the events list during the simulation will be made in this sheet (in this way, the original information are not lost).



	A	B	C	D	E	F
1						
2	Event List					
3	Date	Type	Number	Serial	Dressing Time	
4	6-gen-2000	d	1	11	6	
5	**6-Jan-2000**	s	1	14		
6	**12-gen-2000**	s	1	15		
7	19-gen-2000	s	1	16		
8	2-feb-2000	s	1	17		
9	11-feb-2000	s	1	18		
10	20-feb-2000	d	1	12	8	
11	24-feb-2000	s	1	19		
12	1-mar-2000	d	1	13	9	
13	8-mar-2000	s	1	20		
14	11-mar-2000	d	1	14	4	
15	29-mar-2000	s	1	21		
16	30-mar-2000	d	1	15	6	
17	1-apr-2000	d	1	16	5	
18	12-apr-2000	s	1	22		
19	19-apr-2000	s	1	23		
20	26-apr-2000	d	1	17	7	
21	26-apr-2000	d	1	18	5	
22	26-apr-2000	s	1	24		
23	6-mag-2000	d	1	19	5	
24	10-mag-2000	s	1	25		
25	17-mag-2000	s	1	26		
26	17-mag-2000	s	1	27		
27	17-mag-2000	s	1	28		
28	19-mag-2000	d	1	20	5	
29	24-mag-2000	s	1	29		
30	7-giu-2000	s	1	30		
31	7-giu-2000	s	1	31		
32	15-giu-2000	d	1	21	6	
33	15-giu-2000	s	1	32		
34	15-giu-2000	s	1	33		
35	21-giu-2000	s	1	34		

Fig. 10.21 The rows in black must not be considered during simulation

The simulation starts by inserting the initial data in the initialization window (Figure 10.22). After inserting the first event (grinding wheel S/N 11 of row 4 in Figure 10.21), we proceed to the next one (grinding wheel S/N 14 of row 5 in Figure 10.21 (Figure 10.23). By pressing the “OK” button, the software decides if the grinding wheel must be sent for dressing or kept “on hold” in the warehouse (Figure 10.24).

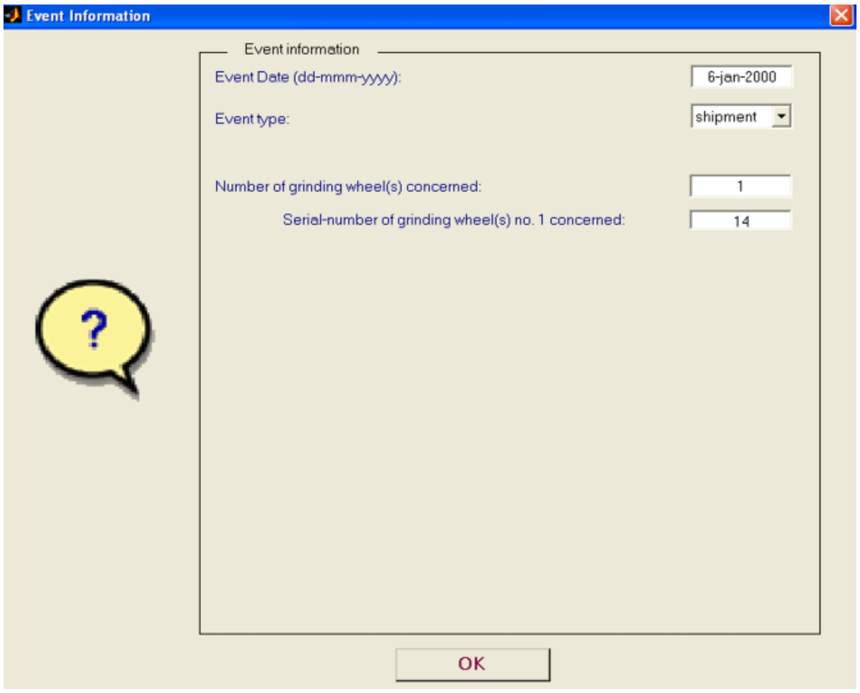


Starting date (dd-mmm-yyyy) : 1-jan-2000

Part number information

Part-number:	U5102N
Current purchase cost (€/unit) :	1200
Current dressing cost (€/unit) :	585
Planned tool demand rate (units/month) :	5.73
Initial on-hand inventory for part-number U5102N (units) :	18
Number of grinding wheels of last year undelivered :	3

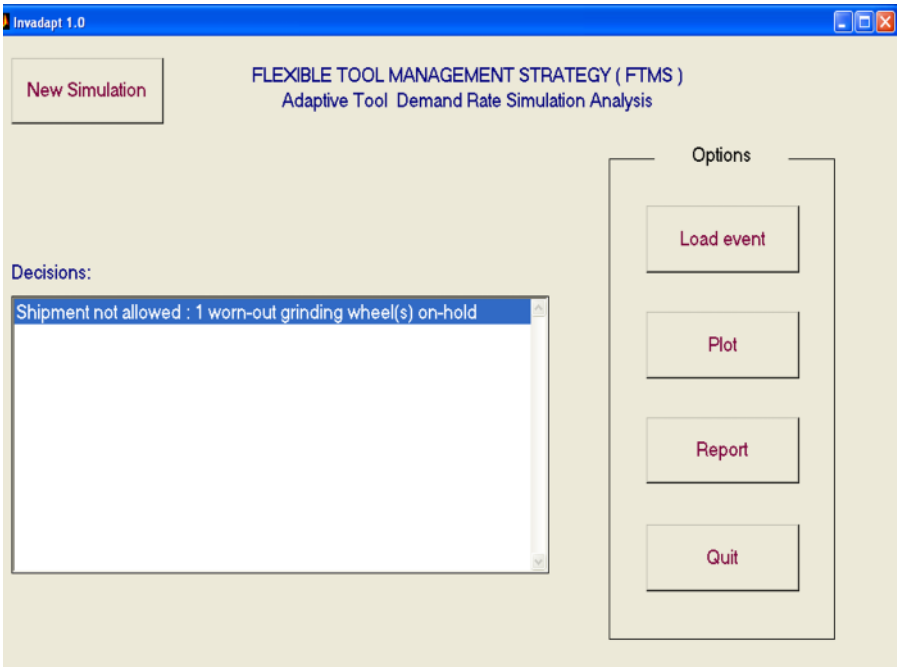
Fig. 10.22 Initialization window for P/N U5102N



Event information

Event Date (dd-mmm-yyyy):	6-jan-2000
Event type:	shipment
Number of grinding wheel(s) concerned:	1
Serial-number of grinding wheel(s) no. 1 concerned:	14

Fig. 10.23 Event information window for the 6 January 2000 event related to CBN grinding wheel S/N 14


Invadapt 1.0
New Simulation
FLEXIBLE TOOL MANAGEMENT STRATEGY (FTMS)
Adaptive Tool Demand Rate Simulation Analysis
Options
Load event
Plot
Report
Quit
Decisions:

Shipment not allowed : 1 worn-out grinding wheel(s) on-hold

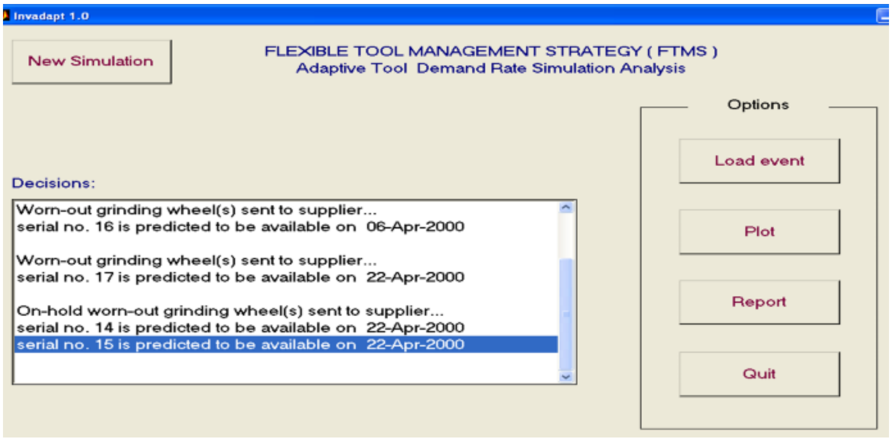
Fig. 10.24 Decisions window for CBN grinding wheel S/N 14

If it is decided to keep grinding wheel 14 "on hold", in the sheet of Figure 10.21, the row related to the delivery event (d in the type column) of grinding wheel 14 (Figure 10.21, row 14 in black) must be cancelled. Likewise, we proceed to the next event observing that also grinding wheel 15 is kept "on hold". The case of the event of 19 January 2000, related to worn-out grinding wheel 16, is different: the program decides to send it for dressing (pointing out the presumed date of availability). In this case the sheet (Figure 10.21) is not modified. We insert the event of 2 February 2000 (grinding wheel 17 of row 8 in Figure 10.21) in which the shipment of grinding wheel 17 occurs, getting the decision shown in Figure 10.25.

The software has decided to send for dressing not only the worn-out grinding wheel (17), but also the two grinding wheels that were "on hold" (14, 15), pointing out their date of availability.

Then the date of actual return grinding wheels 14 and 15 must be calculated: to the date of actual shipment (it does not coincide with the disassembly date) the relative dressing time and the internal time must be added.

The "internal time" is the time interval between the delivery of the grinding wheel from the supplier and the date of assembly: in our case this time is equal to 5 weeks and it is considered constant.


Invadapt 1.0
New Simulation
FLEXIBLE TOOL MANAGEMENT STRATEGY (FTMS)
Adaptive Tool Demand Rate Simulation Analysis
Options
Load event
Plot
Report
Quit
Decisions:

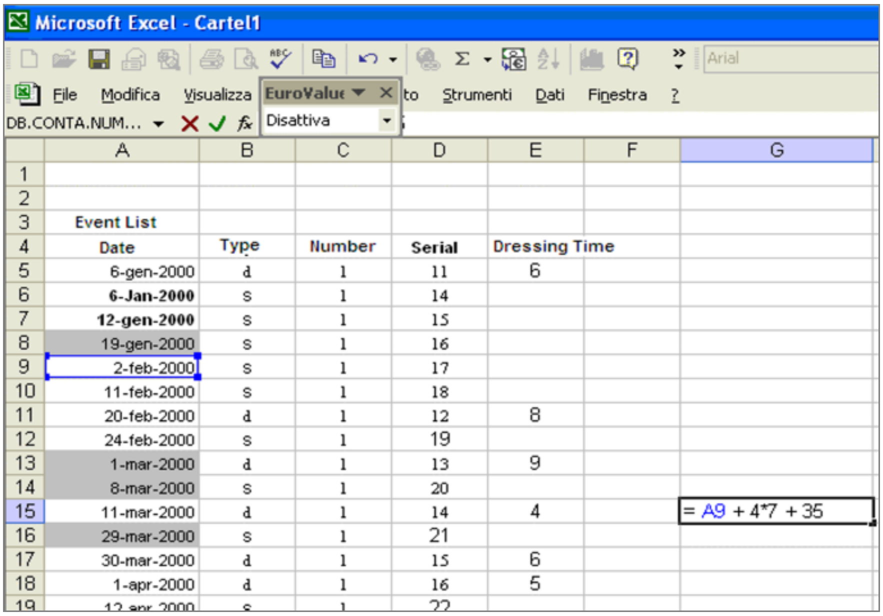
Worn-out grinding wheel(s) sent to supplier...
serial no. 16 is predicted to be available on 06-Apr-2000

Worn-out grinding wheel(s) sent to supplier...
serial no. 17 is predicted to be available on 22-Apr-2000

On-hold worn-out grinding wheel(s) sent to supplier...
serial no. 14 is predicted to be available on 22-Apr-2000
serial no. 15 is predicted to be available on 22-Apr-2000

Fig. 10.25 Window for grinding wheels 14, 15, 16 and 17

For grinding wheel 14: the date of actual shipment coincides with the shipment date of grinding wheel 17 because the two grinding wheels depart simultaneously (the date must be read in column A, row 7 of the file in Figure 10.21: 2 February 2000). To this date, the dressing time related to this grinding wheel retrieved from column E, row 14 of the file in Figure 10.21 (therefore 4) and the internal time equal to 5 weeks, must be added. The obtained date is 5 April 2000 (Figure 10.26).

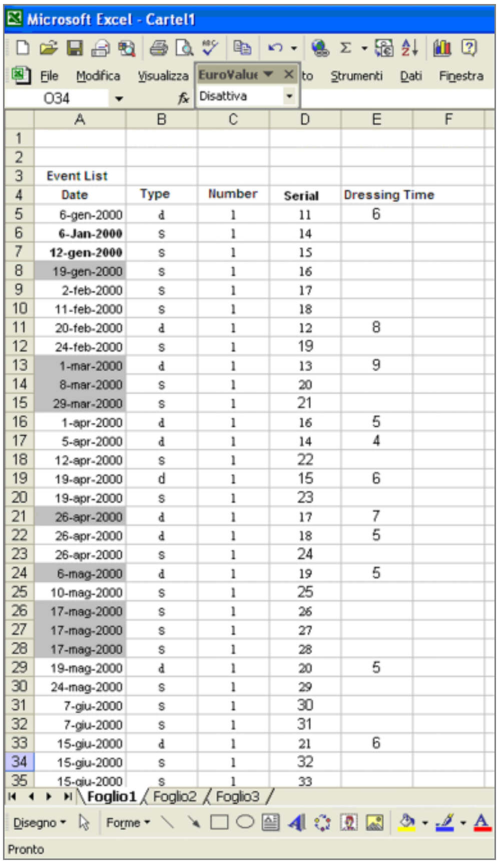


	A	B	C	D	E	F	G
1							
2							
3	Event List						
4	Date	Type	Number	Serial	Dressing Time		
5	6-gen-2000	d	1	11	6		
6	6-Jan-2000	s	1	14			
7	12-gen-2000	s	1	15			
8	19-gen-2000	s	1	16			
9	2-feb-2000	s	1	17			
10	11-feb-2000	s	1	18			
11	20-feb-2000	d	1	12	8		
12	24-feb-2000	s	1	19			
13	1-mar-2000	d	1	13	9		
14	8-mar-2000	s	1	20			
15	11-mar-2000	d	1	14	4		= A9 + 4*7 + 35
16	29-mar-2000	s	1	21			
17	30-mar-2000	d	1	15	6		
18	1-apr-2000	d	1	16	5		
19	12-apr-2000	s	1	22			

Fig. 10.26 Actual return time calculation

For grinding wheel 15: the date of actual shipment coincides with the shipment of grinding wheel 17 (it must be read in column A, row 7 of the file in Figure 10.8). To this date, the dressing time related to this grinding wheel retrieved from column E, row 16 of the file in Figure 10.21 (therefore 6) and the internal time, must be added. The obtained date is 19 April 2000.

In the adaptive list,[5] two rows related to the delivery events of grinding wheels 14 and 15 are added by modifying the historical list of events that coincides with the updated sheet in Figure 10.14. The rows related to this last operation are rows 17 and the 19 of Figure 10.27.



	A	B	C	D	E	F
1						
2						
3	Event List					
4	Date	Type	Number	Serial	Dressing Time	
5	6-gen-2000	d	1	11	6	
6	6-Jan-2000	s	1	14		
7	12-gen-2000	s	1	15		
8	19-gen-2000	s	1	16		
9	2-feb-2000	s	1	17		
10	11-feb-2000	s	1	18		
11	20-feb-2000	d	1	12	8	
12	24-feb-2000	s	1	19		
13	1-mar-2000	d	1	13	9	
14	8-mar-2000	s	1	20		
15	29-mar-2000	s	1	21		
16	1-apr-2000	d	1	16	5	
17	5-apr-2000	d	1	14	4	
18	12-apr-2000	s	1	22		
19	19-apr-2000	d	1	15	6	
20	19-apr-2000	s	1	23		
21	26-apr-2000	d	1	17	7	
22	26-apr-2000	d	1	18	5	
23	26-apr-2000	s	1	24		
24	6-mag-2000	d	1	19	5	
25	10-mag-2000	s	1	25		
26	17-mag-2000	s	1	26		
27	17-mag-2000	s	1	27		
28	17-mag-2000	s	1	28		
29	19-mag-2000	d	1	20	5	
30	24-mag-2000	s	1	29		
31	7-giu-2000	s	1	30		
32	7-giu-2000	s	1	31		
33	15-giu-2000	d	1	21	6	
34	15-giu-2000	s	1	32		
35	15-giu-2000	s	1	33		

Fig. 10.27 Excel work sheet for Figure 10.8 after updating

We proceed in this way for all the other events, observing that the software, on 27 June 2000, decides to purchase a new grinding wheel (it also happens on 19 September and 4October 2000). At the end of the simulation, the obtained list is

[5] Adaptive list: the original file of Figure 10.21 modified following decisions of FTMS.

the list of the events related to the management implemented through the FTMS. The simulated and historical inventory level can be graphically visualized by clicking the "plot" key of Figure 10.25 (Figure 10.28).

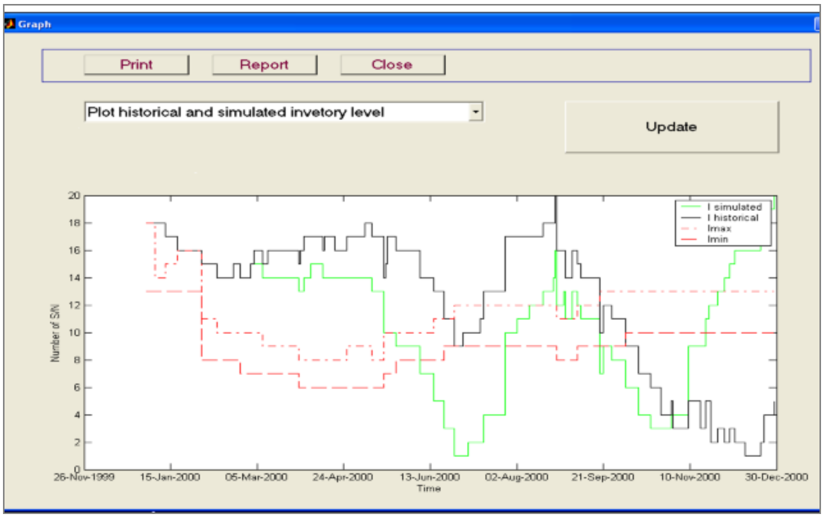


Fig. 10.28 Historical and simulated inventory level trend

This plot allows for the comparison of the simulated inventory level trend *I* of the FTMS management with the historical trend and also reports the two control range limits for the inventory level ($I_{\min}$ and $I_{\max}$). Moreover, the software allows for the visualization of the “adaptive tool demand rate” plot (Figure 10.29).

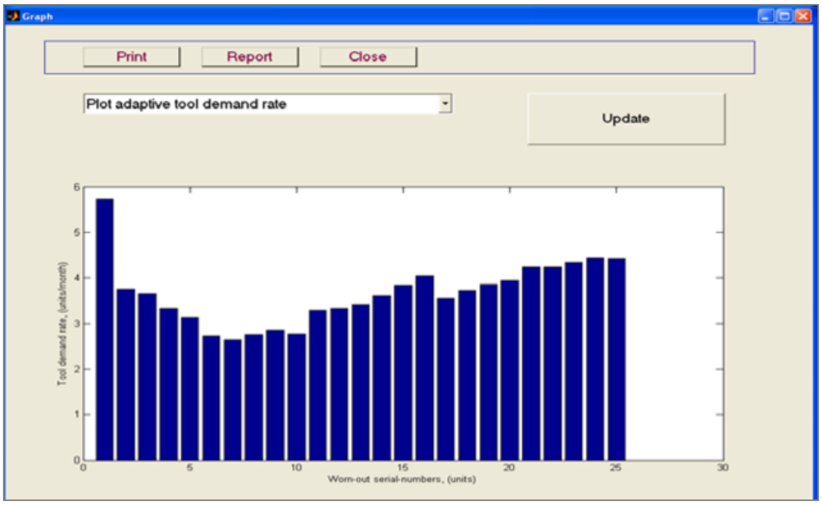

Graph
Print
Report
Close
Plot adaptive tool demand rate
Update
Tool demand rate, (units/month)
0
1
2
3
4
5
6
0
5
10
15
20
25
30
Worn-out serial-numbers, (units)

Fig. 10.29 Adaptive demand rate plot

10.4.3 Simulation Through INVADAPT_NS

To begin the simulation, in the command window of MatLab, the word "INVADAPT_NS" must be typed and the "enter" key pressed. In the command window all information related to the P/N to be simulated must be inputted. The first information to be input is the name of the P/N to simulate, then the date at which the simulation must start: 1 January 2000 (it must be input using the international format 1-jan-2000).

Subsequently, the purchase cost of the CBN grinding wheel must be inputted. In the same way, the following information must be inputted: dressing cost, planned demand rate (given by the ratio among the planned number of pieces to produce *P* and the average life of the grinding wheel *G*), initial on-hand inventory (number of grinding wheels that are available at the start of the simulation), and number of grinding wheels that have been sent to the dressing before the start of the simulation (the starting date inputted in the command window), and that have not been yet delivered. Between an information item and the following the "enter" key must be pressed.

After having inputted all these information items "enter" is pressed once again. The request "event date" appears on screen where the information related to the first event is inputted: the date of the event (in the international format, for example 1 January 2000), the type of event ("shipment" if the grinding wheel is worn out, "delivery" if the dressed grinding wheel is delivered), the number of grinding wheels related to such event.

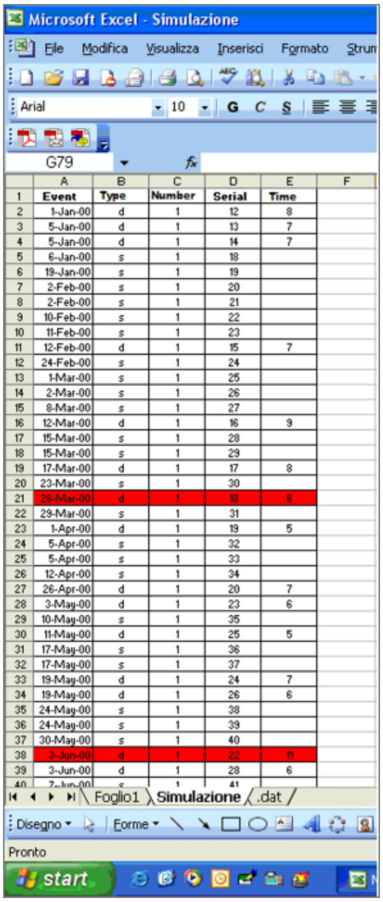
By pressing the "enter" key, a number of rows equal to the number of grinding wheels for the event appear. In these rows the S/Ns must be inputted and, in the case of delivery, an equal number of rows also appear where the corresponding dressing cycle times must be inputted. The system, on the basis of this information, makes the decision related to the management of the worn-out CBN grinding wheels. An example of simulation and the operations that must be carried out to transform the historical list into the adaptive one are reported in the following.

A special sheet must be built by copying the "events list" of Figure 10.30 for convenience. All the changes to the events list during the simulation will be made in this sheet (in this way original information will not be lost). The simulation starts by inputting the initial data in the command window (Figure 10.31).

After inserting the first event (grinding wheel S/N 12 of row 2 in Figure 10.30), the next event is inserted (grinding wheels serial number 13 and S/N 14 of rows 3 and 4 in Figure 10.30). The command window for the 5 January 2000 event related to grinding wheels S/Ns 13, 14 is shown in Figure 10.32.

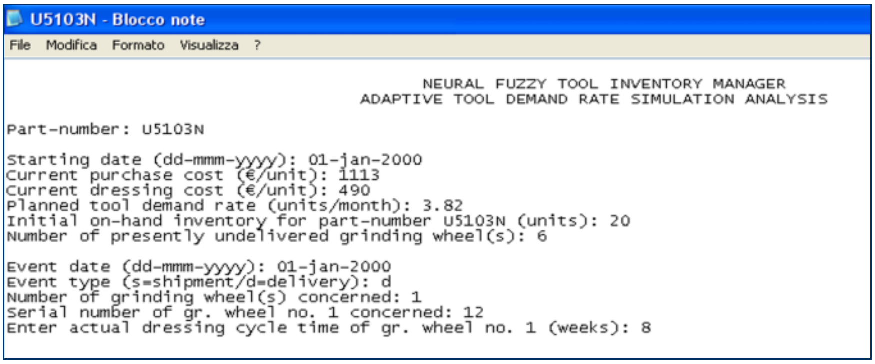
After inserting the second and third events we proceed to the next one (grinding wheel S/N 18 of row 5 in Figure 10.30) (Figure 10.33).

Pressing the "enter" key, the software decides if the grinding wheel must be sent to an external supplier for dressing or kept "on-hold" in the tool warehouse.


Microsoft Excel - Simulazione
File Modifica Visualizza Inserisci Formato Strum
Arial 10 G C S
G79
Foglio1 Simulazione .dat
Disegno Forme
Pronto
start

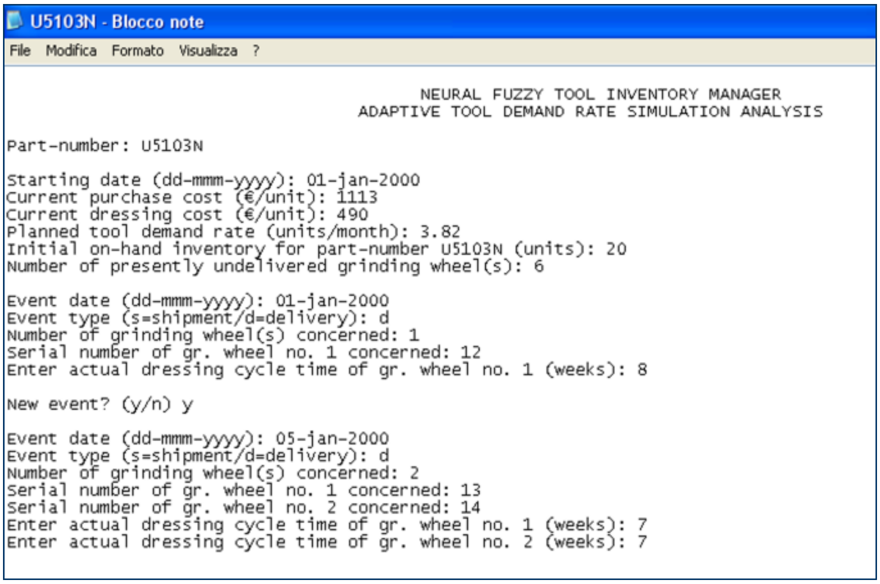
	A	B	C	D	E	F
1	Event	Type	Number	Serial	Time	
2	1-Jan-00	d	1	12	8	
3	5-Jan-00	d	1	13	7	
4	5-Jan-00	d	1	14	7	
5	6-Jan-00	s	1	18		
6	19-Jan-00	s	1	19		
7	2-Feb-00	s	1	20		
8	2-Feb-00	s	1	21		
9	10-Feb-00	s	1	22		
10	11-Feb-00	s	1	23		
11	12-Feb-00	d	1	15	7	
12	24-Feb-00	s	1	24		
13	1-Mar-00	s	1	25		
14	2-Mar-00	s	1	26		
15	8-Mar-00	s	1	27		
16	12-Mar-00	d	1	16	9	
17	15-Mar-00	s	1	28		
18	15-Mar-00	s	1	29		
19	17-Mar-00	d	1	17	8	
20	23-Mar-00	s	1	30		
21	26-Mar-00	d	1	18	6	
22	29-Mar-00	s	1	31		
23	1-Apr-00	d	1	19	5	
24	5-Apr-00	s	1	32		
25	5-Apr-00	s	1	33		
26	12-Apr-00	s	1	34		
27	26-Apr-00	d	1	20	7	
28	3-May-00	d	1	23	6	
29	10-May-00	s	1	35		
30	11-May-00	d	1	25	5	
31	17-May-00	s	1	36		
32	17-May-00	s	1	37		
33	19-May-00	d	1	24	7	
34	19-May-00	d	1	26	6	
35	24-May-00	s	1	38		
36	24-May-00	s	1	39		
37	30-May-00	s	1	40		
38	3-Jun-00	d	1	22	11	
39	3-Jun-00	d	1	28	6	
40	7-Jun-00	s	1	41		

Fig. 10.30 The rows in black must not be considered during simulation



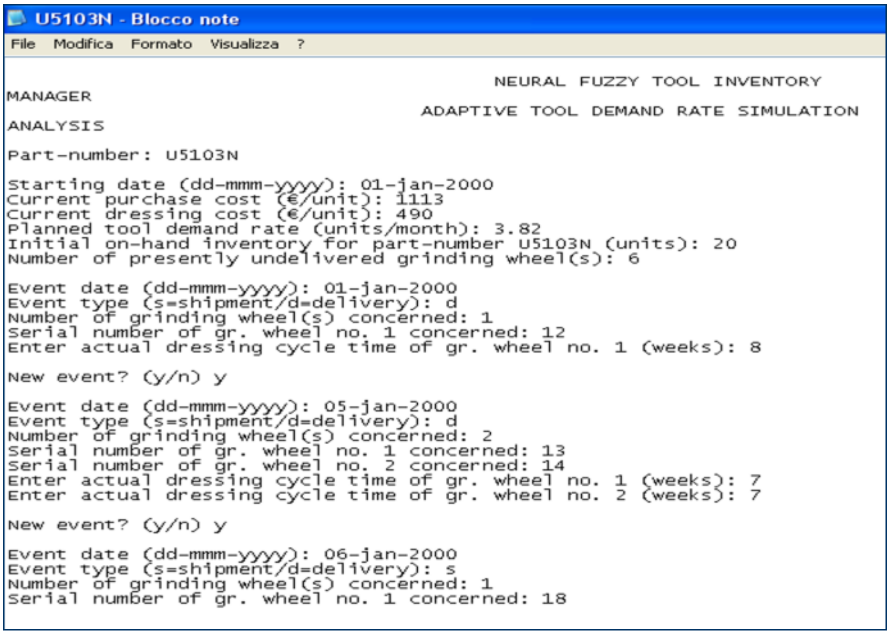
```
                                   NEURAL FUZZY TOOL INVENTORY MANAGER
                              ADAPTIVE TOOL DEMAND RATE SIMULATION ANALYSIS

Part-number: U5103N

Starting date (dd-mmm-yyyy): 01-jan-2000
Current purchase cost (€/unit): 1113
Current dressing cost (€/unit): 490
Planned tool demand rate (units/month): 3.82
Initial on-hand inventory for part-number U5103N (units): 20
Number of presently undelivered grinding wheel(s): 6

Event date (dd-mmm-yyyy): 01-jan-2000
Event type (s=shipment/d=delivery): d
Number of grinding wheel(s) concerned: 1
Serial number of gr. wheel no. 1 concerned: 12
Enter actual dressing cycle time of gr. wheel no. 1 (weeks): 8
```

Fig. 10.31 Command window for the 1 January 2000 event related to CBN grinding wheel S/N 12



```
                                        NEURAL FUZZY TOOL INVENTORY MANAGER
                                 ADAPTIVE TOOL DEMAND RATE SIMULATION ANALYSIS

Part-number: U5103N

Starting date (dd-mmm-yyyy): 01-jan-2000
Current purchase cost (€/unit): 1113
Current dressing cost (€/unit): 490
Planned tool demand rate (units/month): 3.82
Initial on-hand inventory for part-number U5103N (units): 20
Number of presently undelivered grinding wheel(s): 6

Event date (dd-mmm-yyyy): 01-jan-2000
Event type (s=shipment/d=delivery): d
Number of grinding wheel(s) concerned: 1
Serial number of gr. wheel no. 1 concerned: 12
Enter actual dressing cycle time of gr. wheel no. 1 (weeks): 8

New event? (y/n) y

Event date (dd-mmm-yyyy): 05-jan-2000
Event type (s=shipment/d=delivery): d
Number of grinding wheel(s) concerned: 2
Serial number of gr. wheel no. 1 concerned: 13
Serial number of gr. wheel no. 2 concerned: 14
Enter actual dressing cycle time of gr. wheel no. 1 (weeks): 7
Enter actual dressing cycle time of gr. wheel no. 2 (weeks): 7
```

Fig. 10.32 Command window for the 5 January 2000 event related to grinding wheels S/Ns 13, 14



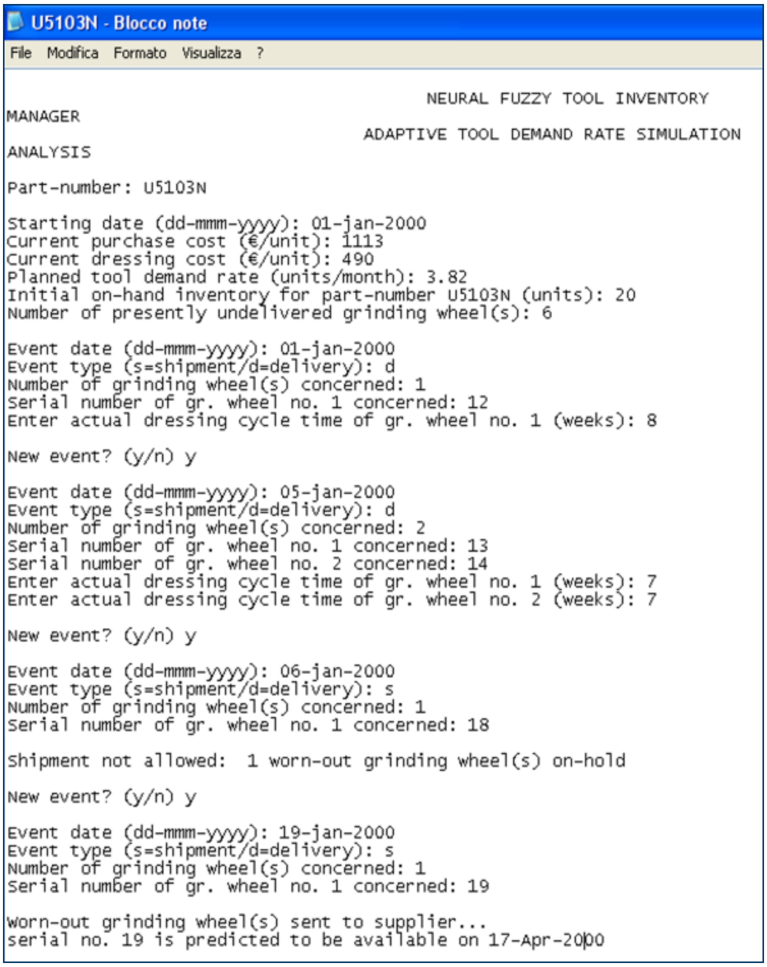
```
                                        NEURAL FUZZY TOOL INVENTORY
MANAGER
                                 ADAPTIVE TOOL DEMAND RATE SIMULATION
ANALYSIS

Part-number: U5103N

Starting date (dd-mmm-yyyy): 01-jan-2000
Current purchase cost (€/unit): 1113
Current dressing cost (€/unit): 490
Planned tool demand rate (units/month): 3.82
Initial on-hand inventory for part-number U5103N (units): 20
Number of presently undelivered grinding wheel(s): 6

Event date (dd-mmm-yyyy): 01-jan-2000
Event type (s=shipment/d=delivery): d
Number of grinding wheel(s) concerned: 1
Serial number of gr. wheel no. 1 concerned: 12
Enter actual dressing cycle time of gr. wheel no. 1 (weeks): 8

New event? (y/n) y

Event date (dd-mmm-yyyy): 05-jan-2000
Event type (s=shipment/d=delivery): d
Number of grinding wheel(s) concerned: 2
Serial number of gr. wheel no. 1 concerned: 13
Serial number of gr. wheel no. 2 concerned: 14
Enter actual dressing cycle time of gr. wheel no. 1 (weeks): 7
Enter actual dressing cycle time of gr. wheel no. 2 (weeks): 7

New event? (y/n) y

Event date (dd-mmm-yyyy): 06-jan-2000
Event type (s=shipment/d=delivery): s
Number of grinding wheel(s) concerned: 1
Serial number of gr. wheel no. 1 concerned: 18
```

Fig. 10.33 Command window for the 6 January 2000 event related to CBN grinding wheel S/N 18

If it is decided to keep the grinding wheel S/N 18 "on hold" in the warehouse, in the sheet of Figure 10.30, the row related to the delivery event (d in the type column) of grinding wheel S/N 18 (Figure 10.30, row 21 in black) must be cancelled.

The event dated 19 January 2000 (related to worn-out grinding wheel 19 of row 6 in Figure 10.30), at which the shipment of grinding wheel S/N 19 occurs, is inputted. The software takes the decision shown in Figure 10.34.

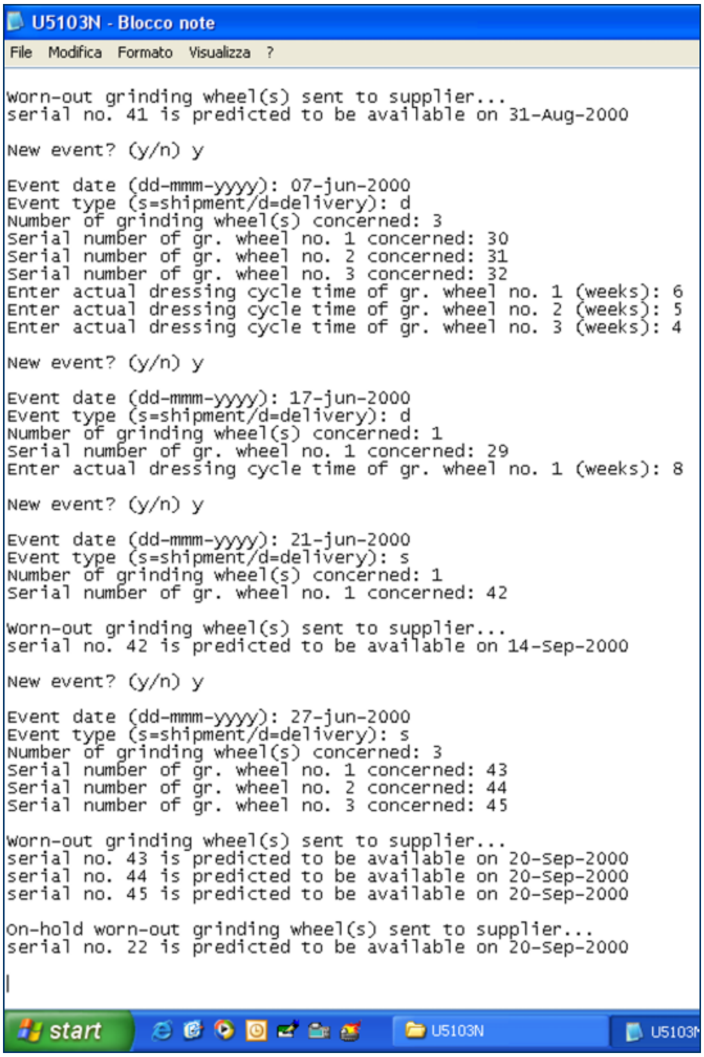
From Figure 10.34, it can be observed that the software has decided to send to an external supplier for dressing grinding wheel S/N 110. In this case the sheet of Figure 10.30 is not modified. We proceed for the next events and can be observed that also grinding wheel 22 is kept "on hold".



```
                                        NEURAL FUZZY TOOL INVENTORY
MANAGER
                                  ADAPTIVE TOOL DEMAND RATE SIMULATION
ANALYSIS

Part-number: U5103N

Starting date (dd-mmm-yyyy): 01-jan-2000
Current purchase cost (€/unit): 1113
Current dressing cost (€/unit): 490
Planned tool demand rate (units/month): 3.82
Initial on-hand inventory for part-number U5103N (units): 20
Number of presently undelivered grinding wheel(s): 6

Event date (dd-mmm-yyyy): 01-jan-2000
Event type (s=shipment/d=delivery): d
Number of grinding wheel(s) concerned: 1
Serial number of gr. wheel no. 1 concerned: 12
Enter actual dressing cycle time of gr. wheel no. 1 (weeks): 8

New event? (y/n) y

Event date (dd-mmm-yyyy): 05-jan-2000
Event type (s=shipment/d=delivery): d
Number of grinding wheel(s) concerned: 2
Serial number of gr. wheel no. 1 concerned: 13
Serial number of gr. wheel no. 2 concerned: 14
Enter actual dressing cycle time of gr. wheel no. 1 (weeks): 7
Enter actual dressing cycle time of gr. wheel no. 2 (weeks): 7

New event? (y/n) y

Event date (dd-mmm-yyyy): 06-jan-2000
Event type (s=shipment/d=delivery): s
Number of grinding wheel(s) concerned: 1
Serial number of gr. wheel no. 1 concerned: 18

Shipment not allowed:  1 worn-out grinding wheel(s) on-hold

New event? (y/n) y

Event date (dd-mmm-yyyy): 19-jan-2000
Event type (s=shipment/d=delivery): s
Number of grinding wheel(s) concerned: 1
Serial number of gr. wheel no. 1 concerned: 19

Worn-out grinding wheel(s) sent to supplier...
Serial no. 19 is predicted to be available on 17-Apr-2000
```

Fig. 10.34 Command window for the 19 January 2000 event related to CBN grinding wheel S/N 19

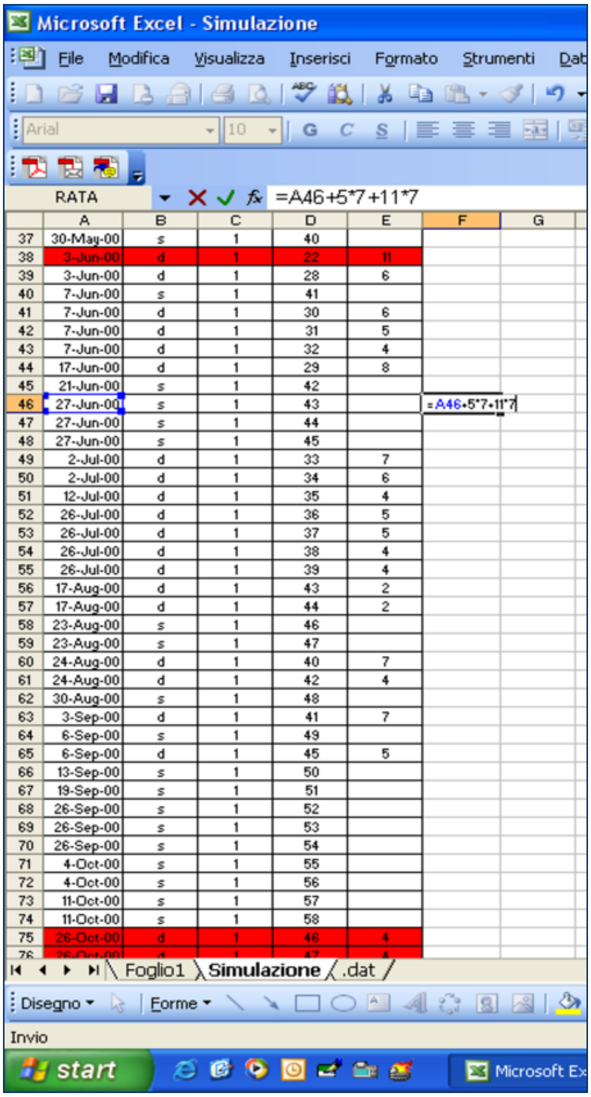
From Figure 10.35, it can be observed that the software, for the events dated 27 June 2000 (related to the worn-out grinding wheel 43, 44, 45 of rows 46, 47, 48 in Figure 10.36) has decided to send to an external supplier not only the worn-out grinding wheels 43, 44, 45, but also the worn-out grinding wheel that was kept "on-hold" (22) pointing out the date at which it is predicted to be available again. We must therefore calculate the date at which grinding wheel 22 will really return: to the date of actual shipment (it does not coincide with the disassembly date) the corresponding dressing cycle time and the internal time must be added.



```
worn-out grinding wheel(s) sent to supplier...
serial no. 41 is predicted to be available on 31-Aug-2000

New event? (y/n) y

Event date (dd-mmm-yyyy): 07-jun-2000
Event type (s=shipment/d=delivery): d
Number of grinding wheel(s) concerned: 3
Serial number of gr. wheel no. 1 concerned: 30
Serial number of gr. wheel no. 2 concerned: 31
Serial number of gr. wheel no. 3 concerned: 32
Enter actual dressing cycle time of gr. wheel no. 1 (weeks): 6
Enter actual dressing cycle time of gr. wheel no. 2 (weeks): 5
Enter actual dressing cycle time of gr. wheel no. 3 (weeks): 4

New event? (y/n) y

Event date (dd-mmm-yyyy): 17-jun-2000
Event type (s=shipment/d=delivery): d
Number of grinding wheel(s) concerned: 1
Serial number of gr. wheel no. 1 concerned: 29
Enter actual dressing cycle time of gr. wheel no. 1 (weeks): 8

New event? (y/n) y

Event date (dd-mmm-yyyy): 21-jun-2000
Event type (s=shipment/d=delivery): s
Number of grinding wheel(s) concerned: 1
Serial number of gr. wheel no. 1 concerned: 42

worn-out grinding wheel(s) sent to supplier...
serial no. 42 is predicted to be available on 14-Sep-2000

New event? (y/n) y

Event date (dd-mmm-yyyy): 27-jun-2000
Event type (s=shipment/d=delivery): s
Number of grinding wheel(s) concerned: 3
Serial number of gr. wheel no. 1 concerned: 43
Serial number of gr. wheel no. 2 concerned: 44
Serial number of gr. wheel no. 3 concerned: 45

worn-out grinding wheel(s) sent to supplier...
serial no. 43 is predicted to be available on 20-Sep-2000
serial no. 44 is predicted to be available on 20-Sep-2000
serial no. 45 is predicted to be available on 20-Sep-2000

on-hold worn-out grinding wheel(s) sent to supplier...
serial no. 22 is predicted to be available on 20-Sep-2000

|
```

Fig. 10.35 Command window for the 27 June 2000 events related to grinding wheels S/Ns 43, 44, 45

The "internal time" is the time interval between the delivery of the grinding wheel from the supplier and the date of assembly: in our case this time is equal to 5 weeks and it is considered constant. Then, the date of actual shipment for grinding wheel 22 coincides with the historical shipment dates of grinding wheels 43, 44, 45. By adding to the date of actual shipment the dressing cycle time related to this grinding wheel (retrieved from column E, row 38 in Figure 10.30 (therefore 11) and the internal time equal to 5 weeks, the obtained date is 17 October 2000 (Figure 10.36).

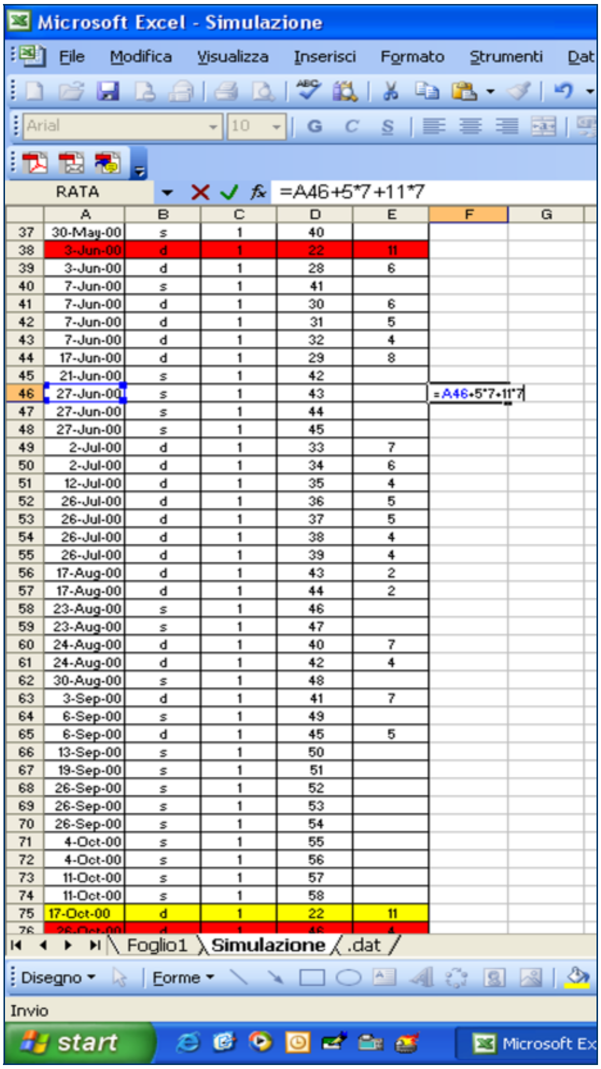


RATA =A46+5*7+11*7

	A	B	C	D	E	F	G
37	30-May-00	s	1	40			
38	3-Jun-00	d	1	22	11		
39	3-Jun-00	d	1	28	6		
40	7-Jun-00	s	1	41			
41	7-Jun-00	d	1	30	6		
42	7-Jun-00	d	1	31	5		
43	7-Jun-00	d	1	32	4		
44	17-Jun-00	d	1	29	8		
45	21-Jun-00	s	1	42			
46	27-Jun-00	s	1	43		=A46+5*7+11*7	
47	27-Jun-00	s	1	44			
48	27-Jun-00	s	1	45			
49	2-Jul-00	d	1	33	7		
50	2-Jul-00	d	1	34	6		
51	12-Jul-00	d	1	35	4		
52	26-Jul-00	d	1	36	5		
53	26-Jul-00	d	1	37	5		
54	26-Jul-00	d	1	38	4		
55	26-Jul-00	d	1	39	4		
56	17-Aug-00	d	1	43	2		
57	17-Aug-00	d	1	44	2		
58	23-Aug-00	s	1	46			
59	23-Aug-00	s	1	47			
60	24-Aug-00	d	1	40	7		
61	24-Aug-00	d	1	42	4		
62	30-Aug-00	s	1	48			
63	3-Sep-00	d	1	41	7		
64	6-Sep-00	s	1	49			
65	6-Sep-00	d	1	45	5		
66	13-Sep-00	s	1	50			
67	19-Sep-00	s	1	51			
68	26-Sep-00	s	1	52			
69	26-Sep-00	s	1	53			
70	26-Sep-00	s	1	54			
71	4-Oct-00	s	1	55			
72	4-Oct-00	s	1	56			
73	11-Oct-00	s	1	57			
74	11-Oct-00	s	1	58			
75	26-Oct-00	d	1	46	4		
76	26-Oct-00	d	1	47	4		

Fig. 10.36 Actual return time calculation

In the adaptive list, one row related to delivery event of grinding wheel 22 is added by modifying the historical list of events that coincides with the updated sheet in Figure 10.37. The row related to this last operation is row 75 of Figure 10.36. We proceed in this way for all the other events. After the last event, at the end of the simulation, the CBN grinding wheel inventory trend is graphically visualized in Figure 10.38. This figure compares the simulated inventory level trend *I* of the FTMS management with the historical one and also reports the two control range limits for the inventory levels ($I_{\min}$ and $I_{\max}$).


Microsoft Excel - Simulazione
File Modifica Visualizza Inserisci Formato Strumenti Dat
Arial 10 G C S
Foglio1 Simulazione .dat
Disegno Forme
Invio
start Microsoft Ex

RATA =A46+5*7+11*7

	A	B	C	D	E	F	G
37	30-May-00	s	1	40			
38	3-Jun-00	d	1	22	11		
39	3-Jun-00	d	1	28	6		
40	7-Jun-00	s	1	41			
41	7-Jun-00	d	1	30	6		
42	7-Jun-00	d	1	31	5		
43	7-Jun-00	d	1	32	4		
44	17-Jun-00	d	1	29	8		
45	21-Jun-00	s	1	42			
46	27-Jun-00	s	1	43		=A46+5*7+11*7	
47	27-Jun-00	s	1	44			
48	27-Jun-00	s	1	45			
49	2-Jul-00	d	1	33	7		
50	2-Jul-00	d	1	34	6		
51	12-Jul-00	d	1	35	4		
52	26-Jul-00	d	1	36	5		
53	26-Jul-00	d	1	37	5		
54	26-Jul-00	d	1	38	4		
55	26-Jul-00	d	1	39	4		
56	17-Aug-00	d	1	43	2		
57	17-Aug-00	d	1	44	2		
58	23-Aug-00	s	1	46			
59	23-Aug-00	s	1	47			
60	24-Aug-00	d	1	40	7		
61	24-Aug-00	d	1	42	4		
62	30-Aug-00	s	1	48			
63	3-Sep-00	d	1	41	7		
64	6-Sep-00	s	1	49			
65	6-Sep-00	d	1	45	5		
66	13-Sep-00	s	1	50			
67	19-Sep-00	s	1	51			
68	26-Sep-00	s	1	52			
69	26-Sep-00	s	1	53			
70	26-Sep-00	s	1	54			
71	4-Oct-00	s	1	55			
72	4-Oct-00	s	1	56			
73	11-Oct-00	s	1	57			
74	11-Oct-00	s	1	58			
75	17-Oct-00	d	1	22	11		
76	26-Oct-00	d	1	46	4		

Fig. 10.37 Excel work sheet of Figure 10.30 after updating

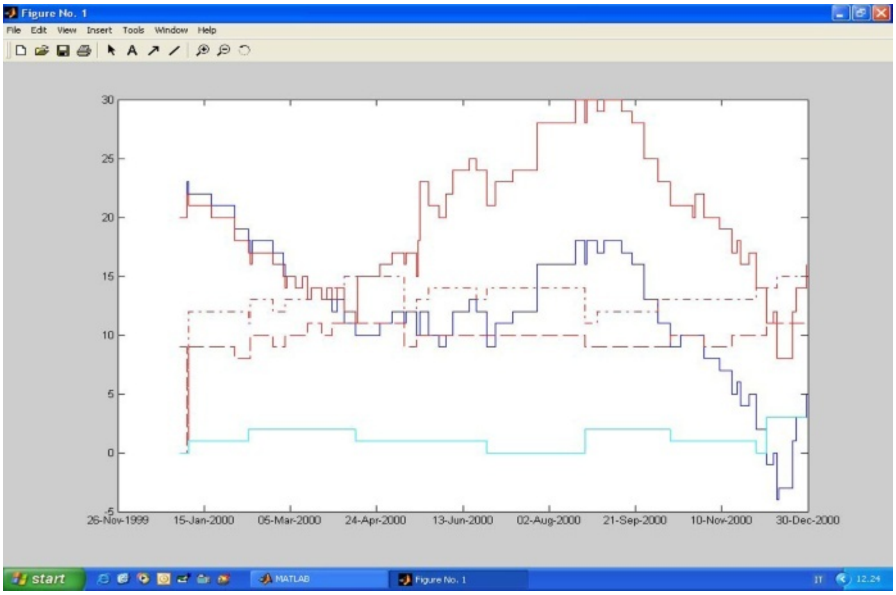


Fig. 10.38 CBN grinding wheel inventory level trend

The software also visualizes the plot of the “tool supply cost” shown in Figure 10.39.

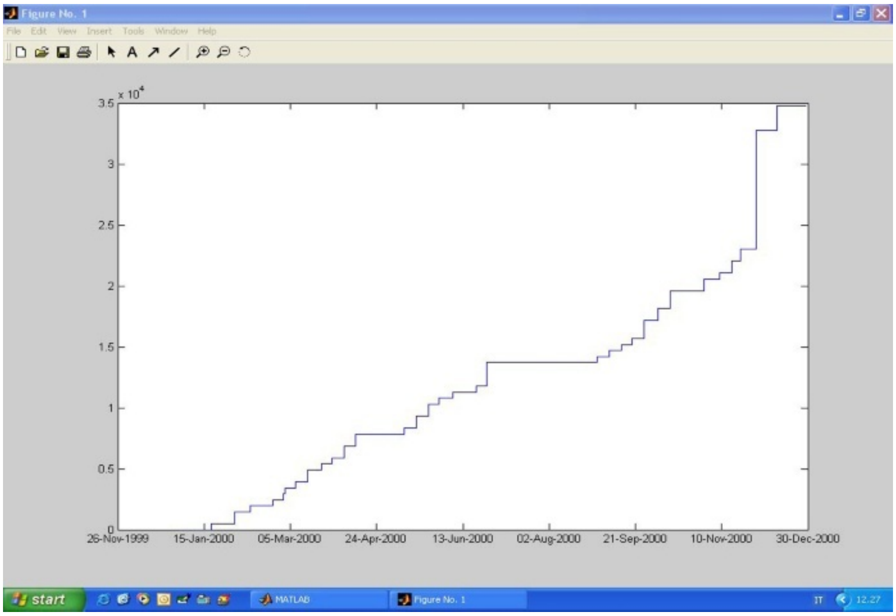


Fig. 10.39 Tool supply cost plot

Moreover the software visualizes the plot of the "adaptive tool demand rate" values reported in Figure 10.40. We proceed in this way for all the other events. At the end of the simulation, the obtained list is the list of the events related to the management implemented through the FTMS. After the last event, at the end of the simulation, the trend of the grinding wheels inventory is graphically visualized in Figure 10.37. This figure compares the simulated inventory level trend *I* of FTMS management with the historical one and also gives two control range limits for the inventory levels ($I_{\min}$ and $I_{\max}$).

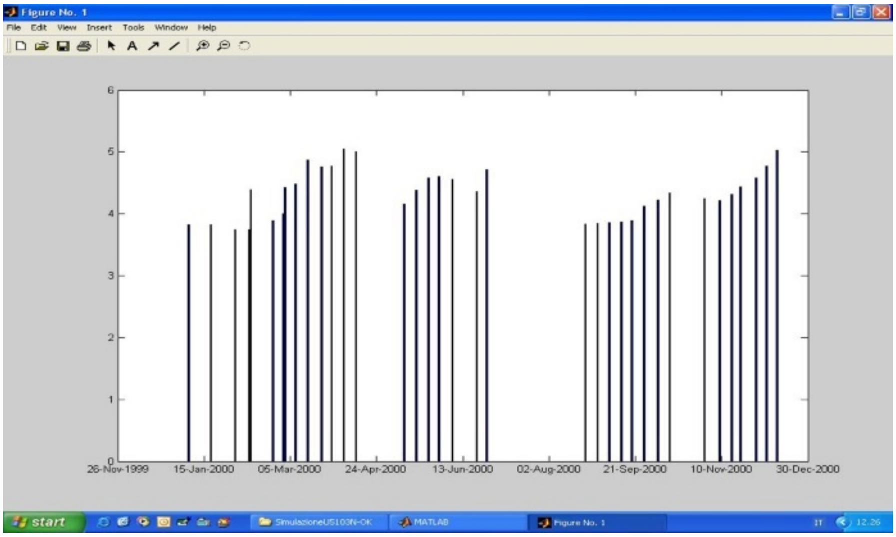


Fig. 10.40 Adaptive tool demand rate plot

10.5 Test Case Applications

The flexible tool management of P/N M3941142-1 was simulated as a test case application of the three FTMS versions using one-year historical tool management data.

In Figure 10.41, the historical and the simulated inventory level trends are reported *versus* time, for the reference period of one year, with the indication of the historical tool supply cost and the cost variation of the simulated trend, for the INVADAPT (Figure 10.41a), I-FUTURE (Figure 10.41b) and the INVADAPT_NS (Figure 10.41c) paradigms.

It is worth recalling that the inventory level historical trend is the result of the tool management activity of experienced staff. Though this management approach based on expert knowledge presents the disadvantage of being dependent on skilled staff whose experience is acquired during years, it is robust and reliable and can be utilized as a reference for assessing intelligent computation procedures of tool management, operating without the support of human experts. From Figure 10.41, it can be observed that the FTMS approaches yield a notable economy over the tool management historical cost for P/N M3941142-1.

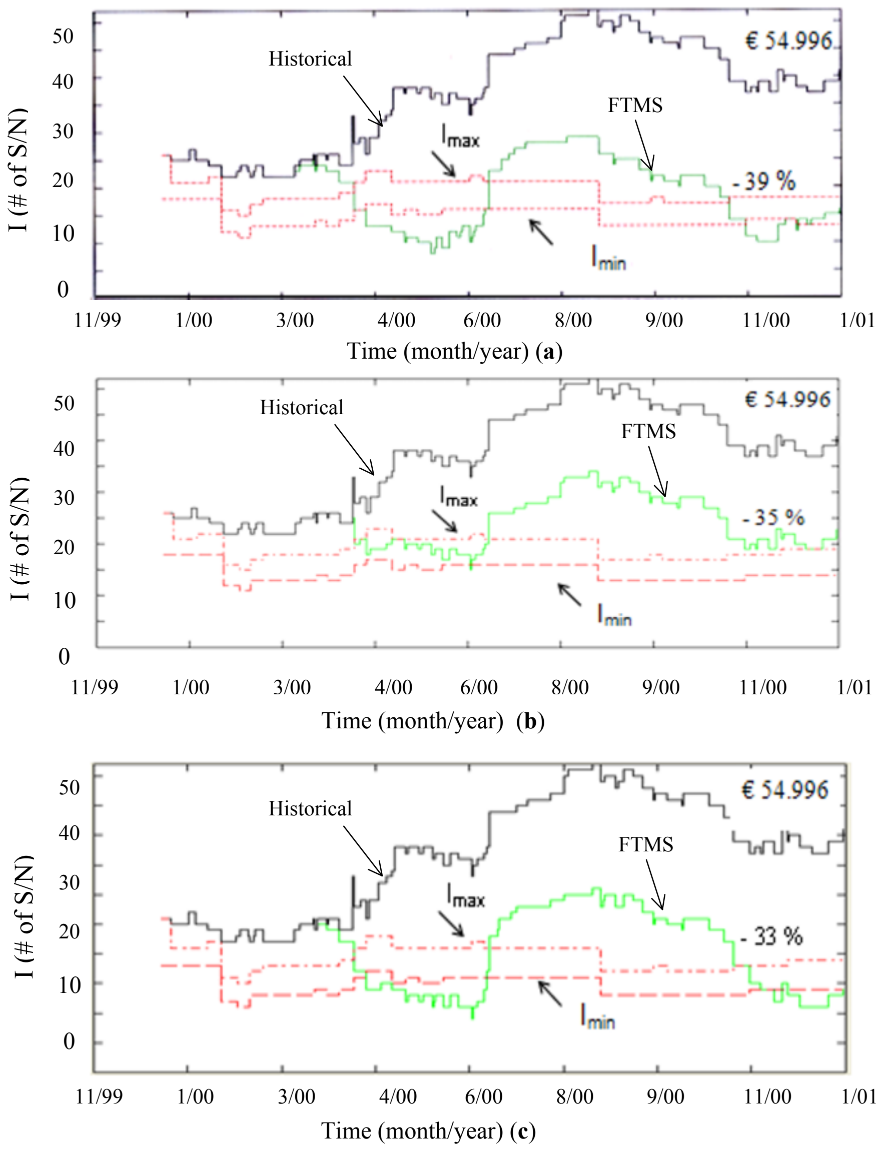


Fig. 10.41 Historical and FTMS simulated on-hand inventory level, *I*, *vs*. time (one year), for P/N M3941142-1: **a** INVADAPT paradigm, **b** I-FUTURE paradigm, **c** INVADAPT_NS paradigm

In Table 10.1, the yearly tool supply cost for all FTMS paradigms are reported for comparison with the traditional tool management: a 39% economy is obtained for INVADAPT, a 35% and a 33% saving is achieved for I_FUTURE and INVADAPT_NS respectively. The significant cost reduction obtained with the FTMS approaches in comparison with the traditional tool management is due to the fact that the simulated on-hand inventory level is higher than the $I_{\max}$ limit for long time intervals during the tool management period, therefore avoiding useless dressing operations and tool purchases.

For the FTMS procedures, the initial tool demand rate value P/G = 8.1 units/month is higher than the mean adaptive demand rate $\overline{d}_R$ after the first month of tool management (see Table 10.2). This generates an initial drop of the $I_{\max}$ value, the more significant for the INVADAPT approach where $\overline{d}_R = 0.78$, P/G = 6.3 units/month, accounting for the higher economy achieved in comparison with the I-FUTURE approach where $\overline{d}_R = 0.79$, P/G = 6.4 units/month and INVADAPT_NS approach where $\overline{d}_R = 0.80$, P/G = 6.5 units/month. The historical trends do not take into consideration the demand rate variation during tool management and CBN grinding wheels keep being sent out for dressing or are newly purchased, thus maintaining a high on-hand inventory level that generates superfluous capital investment by seeking excessive stock-out risk protection.

Table 10.1 Yearly tool supply cost for test case P/N M3941142-1

P/N	Scenario	Supply cost	Cost variation	Purchased tools	Dressed tools
M3941142-	Historical	€ 54.996	–	18	73
	Simulation INVADAPT	€ 33.324	–39%	3	55
	Simulation I-FUTURE		–35%	1	68
	Simulation INVADAPT_NS	€ 36.660	–33%	0	65

Table 10.2 Initial tool demand rate, P/G, and average demand rate, $\overline{d}_R$, during the tool management period for the test case P/N M3941142-1

P/N	Initial demand rate P/G (units/month)	Average adaptive demand rate $\overline{d}_R$ (units/month)
M3941142-1	8.1	6.3
		6.4
		6.5

10.6 Conclusions and Future Work

In this work, tool inventory management simulations were carried out using diverse FTMS paradigms integrated in a MATMS for automatic tool procurement, and the obtained results were compared with reference to real industrial cases.

The MATMS operates in the framework of a negotiation-based multiple-supplier network where a turbine blade producer (customer) requires dressing jobs on worn-out CBN grinding wheels for nickel alloy turbine blade manufacturing from different tool manufacturers (suppliers). The FTMS is designed as a domain-specific problem solving function of the MATMS intelligent agent whose task is the optimum tool inventory sizing and control of CBN grinding wheels: the RA. The latter performs its activities on the basis of running production plans and dressing cycle time predictions, utilizing the FTMS paradigm whereby the P/N tool inventory size varies freely within a real-time variable control range to achieve optimal tool inventory sizing and control. A test case application of three FTMS paradigms based on tool management decision making with reference to the value of a future inventory level $I_{\text{fut}}(t + T_{\text{rt}})$ were presented to illustrate and assess the new FTMS performance *versus* traditional tool management based on human expert knowledge.

By comparing the historical and the FTMS simulated inventory level trends for a test case P/N using the three FTMS approaches, a notable cost reduction was obtained with all FTMS methods that take into account the tool demand rate variations during the tool management execution, preventing unnecessary dressing operations or tool purchases.

The development of the research activity will aim at the generalization of the management approach to the inventory control applications other than CBN grinding wheel tool management.

Acknowledgements This research work was carried out with support from the EC FP6 NoE on Innovative Productions Machines and Systems – I*PROMS. The Laboratory for Advanced Production Technology (LAPT) is gratefully acknowledged for making available to this project work its custom-made proprietary software packages MATMS and FTMS.

References

Fox MS, Barbuceanu M, Teigen R (2000) Agent-oriented supply chain management. Int. J. Flex. Manuf. Syst. 12:165–188

Jang JSR (1993) ANFIS: Adaptive-network-based fuzzy inference system. IEEE Trans. Syst. Man Cyber. 23(3):665–685

Jang JSR (1995) Neuro-fuzzy modelling and control. In: Proceedings of the IEEE 83:378–405

Teti R, D’Addona D (2003a) Agent-based multiple supplier tool management system. In: Prroceedings of the 36th CIRP Int. Sem. on Manuf. Syst. – ISMS 2003, Saarbrücken, 3–5 June, pp. 39–45

Teti R, D'Addona D (2003b) Grinding wheel management through neuro-fuzzy forecasting of dressing time. Ann. CIRP 52(1):407–410

Teti R, D'Addona D (2003c) Multiple supplier neuro-fuzzy reliable delivery forecasting for tool management in a supply network. In: Proceedings of the 6th AITEM Conf., Gaeta, 8–10 September

Teti R, D'Addona D (2004) Adaptive NF tool delivery forecasting for flexible tool management in a supply network. In: Proceedings of the 5th Int. Workshop on Emergent Synthesis – IWES '04, Budapest, 24–25 May

Teti R, D'Addona D, Segreto T (2004) Flexible tool management in a multi-agent modelled supply network. In: Proceedings of the 37th CIRP Int. Sem. on Manuf. Syst. – ISMS, Budapest, 19–21 May, pp. 351–356

Sycara KP (1998) Multi-Agent Systems. AI Magazine, Summer, pp. 79–92

Van Hentenryck (1999) The OPL optimisation programming language. MIT, Boston

Yuan Y, Liang TP, Zhang JJ (2001) Using agent technology to support supply chain management: potentials and challenges. MG De Groote School of Business, Working Paper No. 453

Chapter 11
Intelligent Integrated Maintenance Policies for Manufacturing Systems

N. Rezg and S. Dellagi

Abstract Nowadays, the fierce competition between enterprises has led many of them to revise their maintenance and production or service strategies. Satisfying customer demand in a timely fashion has become difficult due to the demand's random nature, a problem compounded by machine failures and low system availability. Much of the literature addressing this situation describes models and methods to optimize maintenance policies. By contrast, the research presented in this chapter is geared toward the development of a set of intelligent maintenance policies, which optimize integrated maintenance, inventory and production elements while satisfying random demand over future periods. This is a complex task due to the various uncertainties involved, which are due to external or internal factors. For instance, the variations in the environmental and operational conditions can be considered as external factors whereas the variations in the material availability can be considered as internal factors. The approach described in this chapter demonstrates the significant influence these factors have on the system failure rate, which is in turn important in the determination of an optimal intelligent maintenance strategy. We also emphasize that, on the one hand, high machine availability is an underlying assumption of just-in-time production control policies, and on the other hand, the traditional approach, which dissociates maintenance and production, is no longer satisfactory.

Keywords Preventive maintenance (PM), failure rate, simulation, optimization, reliability, availability, sequential PM, periodic PM

11.1 Introduction

Problems treating jointly maintenance and production plans have recently attracted the attention of several researchers. The basic thrust of our research is in

N. Rezg (✉) and S. Dellagi
INRIA Nancy Grand Est /LGIPM – University of Metz, Ile du Saulcy, 57000 Metz, France
e-mail: rezg@univ-metz.fr

devising an intelligent aggregate optimal production and maintenance plan, which minimizes the total cost of production, inventory and maintenance, and satisfies the random demand over future periods. The complexity of this task stems from the various uncertainties associated with the decision process, due to external and internal factors. While the randomness of the demand, which inhibits knowledge of the demand behaviour during future periods, can be considered as an external factor, the variation in material availability can be considered as an internal factor. An intelligent optimal maintenance strategy considering the deterioration of the manufacturing system according to the production rate was studied by Hajej *et al.* (2009). On the other hand Silva and Cezarino (2004) considered a chance-constrained stochastic production-planning problem under the assumption of imperfect information on the inventory variables.

The traditional maintenance policies proposed in the literature mainly seek the critical age of a machine or a set of machines at which preventive maintenance (PM) is to be carried out. These policies are based on models describing the deterioration law of the equipment. Valdez-Flores and Feldman (1989) distinguished four classes of models: inspection models, models of minimal repair, shock models and replacement models. The cost/time of maintenance/repair is assumed to be known and consequently the impact of a maintenance/failure cannot be analysed easily. Under these conditions, it can be shown that the optimal policy is one of critical age. In this context, Wang and Pham (1996) suggested that maintenance can be classified according to the degree to which the operating conditions of an item are restored by maintenance. They proposed four categories: perfect repair or perfect maintenance, minimal repair or minimal maintenance, imperfect repair or imperfect maintenance and worse repair or worse maintenance.

Obviously, maintenance becomes even more important in a just-in-time production environment, which requires the uninterrupted availability of machines. An integrated approach of maintenance and production control hence becomes necessary. In this context, Buzacott and Shanthikumar (1993) proved the importance of the choice of the maintenance policy for the minimization of the total cost, while Cheung and Hausmann (1997) considered the simultaneous optimization of the safety stock and the maintenance policy of the critical age type. Rezg *et al.* (2004) tackled the global optimization of PM and inventory costs in a production line made up of *N* machines. In the same context Rezg *et al.* (2008) presented a mathematical model and a numerical procedure, which determine a joint optimal inventory control, and age-based preventive maintenance policy for a randomly failing production system. Maintenance/production strategies taking into account a subcontractor are considered by Dellagi *et al.* (2007).

Traditional PM models assume that the action is perfect, i.e., it restores the system to the as-good-as-new state. However, it is more realistic to assume imperfect PM actions, where the system following the PM activity is in a better condition than before, but cannot be considered as new. Many models have been developed to represent the effect of an imperfect maintenance. An early such model, developed by Brown and Proschan (1983), assumes that the PM is equivalent to a replacement with probability p and minimum reparation with a probability $(1 - p)$. Malik (1979) proposed a reduction of the hazard rate after PM by means of an age reduction factor, while Lie and Chun (1986) introduced an improvement factor in the hazard rate after maintenance.

The choice of a PM policy is an important step in the planning of maintenance activities as it specifies how these actions must be scheduled. Barlow and Hunter (1960) considered two types of PM policies: periodic and sequential. For a periodic PM, the system is maintained preventively at times kx (x = constant; $k = 1, 2, \ldots, N$) and is replaced at the Nth PM. It is assumed that the system undergoes only minimal repair at failure, and hence, the hazard rate remains unchanged by any repair between PM. Sequential PM is based on the same principle as the periodic PM except that the intervals between PM activities do not have the same fixed length. The periodic PM is easier to program, but the sequential PM is more representative of reality. Indeed, for a system whose hazard rate increases with time, when the system ages, the arrival rate of failures increases, so that it is advisable to reduce PM intervals over time.

The majority of contributions that are found in the PM literature assume that the time horizon is infinite. Among the few articles focusing on finite time horizons, Dedopulos and Smeers (1998) considered a single unit working in a continuous mode of operation. Based on an age reduction approach, they searched for the optimal number of PM activities to be scheduled with the goal of maximizing profit.

Another observation that can be made is that almost all maintenance models that are found in the literature imply that the system is always maintained under fixed conditions. For these models, the failure law, generally obtained by statistical test conducted under nominal conditions in a laboratory, remains the same throughout the system's life. To represent the effects of random environmental conditions, Özekici (1995) proposed to take an intrinsic age of the system rather than the actual age. Martorell *et al.* (1999) used models of accelerated life to accommodate operating conditions, which may vary during the life cycle of the device. The system does not deteriorate evenly and must achieve an ordered finite set of missions, each of which is carried out under different operational conditions.

It goes without saying that new intelligent maintenance strategies of the type developed by the Center for Intelligent Maintenance Systems (IMS), are urgently needed. The vision of IMS is to enable products and systems to achieve and sustain near-zero breakdown performance, and ultimately transform the traditional maintenance practices from "fail and fix" to "predict and prevent" methodology. More precisely, the key to their approach is to transform machine condition data to useful information for improved productivity and asset utilization. In fact, IMS is focused on frontier technologies in embedded and remote monitoring, prognostics technologies, and intelligent decision support tools and has coined the trademarked Watchdog Agent® prognostics tools and Device-to-Business (D2B) infotronics platform for e-maintenance systems.

This chapter reports on two studies dealing with complex reliability issues and investigates new intelligent integrated maintenance and production or service strategies. The first study, motivated by the lack of tools to improve the performance of production systems in the presence of maintenance activities, presents the development of an intelligent optimal joint production and maintenance plan under random demand. Furthermore, it is assumed that the equipment deteriorates according to the production rate, a factor seldom considered in the literature. Our method begins by solving a constrained stochastic optimization problem yielding a production plan, which minimizes the average total holding and production costs. Next, using the optimal production plan obtained and its influence on the manufacturing system failure rate, an optimal maintenance schedule which minimizes the total expected maintenance cost is established by analytical study and intelligent means.

The second study introduces an adaptive failure law, which can vary over time depending on the missions to achieve. Assuming this deterioration law and a finite time horizon *H*, the purpose of this study is to develop a periodic PM policy characterized by a set of maintenance actions at fixed intervals with renewal at time *H*. The problem is to determine intelligently the number of PM actions which must be performed so as to minimize a cost criterion. Moreover, the cost and the PM duration depend on a coefficient of quality. Thereafter, a constraint is added in order to ensure a minimal availability. Finally, the proposed approaches in both studies are illustrated via numerical examples.

The rest of the chapter is organized as follows. Sect. 11.2 describes the optimal intelligent integrated maintenance policy considering the influence of the production plan on the deterioration of the manufacturing system. Sect. 11.3 presents the developed intelligent periodic preventive maintenance policy in finite horizon with an adaptive failure law. Finally, Sect. 11.4 concludes the chapter with some directions for future work.

11.2 Optimal Maintenance Policy Considering the Influence of the Production Plan on the Deterioration of the Manufacturing System

11.2.1 Problem Statement

In this first study, we are concerned with the production planning problem of a manufacturing system composed of a single machine which produces a single product whose demand *d*, characterized by a normal distribution with a mean and standard deviation respectively denoted by $\hat{d}$ and σ_d^2, is to be satisfied at minimum cost. The problem is illustrated in Figure 11.1. The machine is subject to random failure, its deterioration being described by the probability density function of time to failure $f(t)$, whose failure rate $\lambda(t)$ increases with time and according to the production rate $u(t)$. Failures of the machine can be prevented by appropriate PM measures.

Our first objective is to establish an economical production plan, following which an intelligent technique is used in order to determine the optimal PM period. The use of this two-stage approach is justified by the fact that the production rate has an impact on the failure rate of the machine.

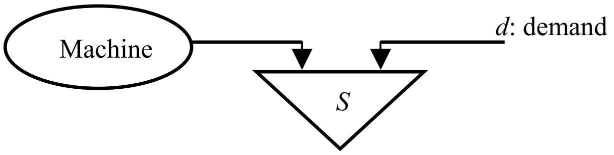


Fig. 11.1 Structure of the studied manufacturing system

11.2.2 Problem Formulation

1.2.2.1 Notation

C_{pr}:	unit production cost
C_{s}:	holding cost of a product unit during the period k
C_{λ}:	penalty degradation cost
CRM:	minimal repair cost
$f(t)$:	probability density function of time to failure for the machine
H:	finite production horizon
c_2:	PM action cost
c_1:	corrective maintenance action cost
mu:	monitory unit
$R(t)$:	reliability function
$s(k)$:	stock level at the end of the period k
$U_{\max}$:	maximal production rate
$u(k)$:	production rate for period k
Z:	total expected cost including production and inventory over a finite horizon H
C:	total maintenance cost expected per time unit

$U = (u(1), u(2), \ldots, u(k), \ldots, u(h))$: production rates vector

α:	probabilistic index; related to the customer satisfaction
Δt:	production period length

Int $(T/\Delta t)$: floor function of $T/\Delta t$ (greatest integer smaller than $T/\Delta t$)

11.2.2.2 Production Policy

This subsection formulates a linear stochastic optimal control problem under a threshold stock level constraint. The objective is to minimize the expected production and holding costs over a finite time horizon [0, *H*] under the following assumptions:

- The horizon *H* is divided equally into *h* periods.
- The demand must be satisfied in each period.
- Storage and production costs C_{pr}, C_s are known and constant.
- The demand standard deviation $\sigma_d(k)$ and the demand mean $\hat{d}(k)$ for each period *k* are known and constant.

Formally, the problem can be expressed as follows:

$$U^* = \min_U (Z(U)).$$

The system model is the state equation specifying the stock level, where S_0 is a given initial stock.

$$E\{s(k+1)\} = E\{s(k)\} + u(k) - E\{d(k)\}, \tag{11.1}$$

$s(0) = s_0.$

The expected production and holding costs for period *k* is given by:

$$z_k(u(k),\ s(k)) = C_s \times E\left\{(s(k))^2\right\} + C_{pr} \times (u(k))^2. \tag{11.2}$$

Remark 11.1: *The use of quadratic costs allows penalizing both excess and shortage of inventory. Therefore, the total expected cost of production and inventory over h periods is expressed by:*

$$Z(u) = \sum_{k=0}^{h-1} z_k(u(k),\ s(k))$$

$$\Rightarrow Z(u) = C_s \times E\left\{\left(s(h)^2\right)\right\} + \sum_{k=0}^{h-1}\left[C_s \times E\left\{\left(s(k)^2\right)\right\} + C_{pr} \times \left(u(k)^2\right)\right]. \tag{11.3}$$

Remark 11.2: $C_{pr} \times (U(h))^2$ *is not included in Eq.11.3 since we do not consider production at the end of the horizon H.*

Thus our problem becomes:

$$\min_u \left[C_s \times E\left\{\left(s(h)^2\right)\right\} + \sum_{k=0}^{h-1}\left[C_s \times \left\{\left(s(k)^2\right)\right\} + C_{pr} \times \left(u(k)^2\right)\right]\right]. \tag{11.4}$$

subject to:

$$E\{s(k+1)\} = E\{s(k)\} + u(k) - E\{d(k)\}, \qquad k = 0,1,\dots, h-1,$$

under the following constraints:

$$\text{Prob}\left[\hat{s}(k+1) \geq 0\right] \geq \alpha, \qquad k = 0,1,\dots, h-1,$$

$$0 \leq u(k) \leq U_{\max}, \qquad k = 0,1,\dots, h-1,$$

where $\hat{s}(k+1)$ is the mean stock level at the end of period (k+1).

Figure 11.2 describes the system dynamic in discrete time.

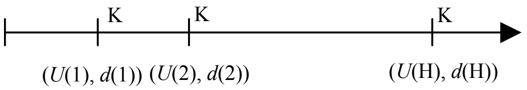


Fig. 11.2 System dynamic in discrete time

We can rewrite the expected value formulation of the production and inventory cost as the following deterministic equivalent:

$$Z(u) = C_s \times \left(\hat{s}(h)^2\right) + \sum_{k=0}^{h-1}\left[C_s \times \hat{s}(k)^2 + C_{pr} \times u(k)^2\right] + C_s \times (\sigma_\text{d})^2 \times \frac{h \times (h-1)}{2}. \tag{11.5}$$

For its part, the probabilistic service level constraint is transformed into a deterministic equivalent constraint by specifying certain minimum cumulative production quantities that depend on the service level requirements.

Lemma 11.1: $\text{Prob}\left(\hat{s}(k+1) \geq 0\right) \geq \alpha \quad \Rightarrow \quad \left(u(k) \geq u_\alpha\left(s(k), \alpha\right)\right)$

where $U_\alpha\left(s(k), \alpha\right) = V_{d,k} \times \phi_{d,k}^{-1}(\alpha) + \hat{d}(k) - \hat{s}(k)$ *and* $\phi_{d,k}$ *is the Gaussian distribution function with mean* $\hat{d}_k$ *and finite variance*

$$\operatorname{Var}(d_k) = V_{d,k} \geq 0 \tag{11.6}$$

where $\phi_{d,k}^{-1}$ *corresponds to the inverse distribution function.*

Proof: $\hat{s}(k+1) = \hat{s}(k) + u(k) - d(k)$

$\Rightarrow \operatorname{Prob}\left(\hat{s}(k+1) \geq 0\right) \geq \alpha$

$\Rightarrow \operatorname{Prob}\left(\hat{s}(k) + u(k) - d(k) \geq 0\right) \geq \alpha$

$\Rightarrow \operatorname{Prob}\left(\hat{s}(k) + u(k) \geq d(k)\right) \geq \alpha$

$\Rightarrow \operatorname{Prob}\left(\hat{s}(k) + u(k) - \hat{d}(k) \geq d(k) - \hat{d}(k)\right) \geq \alpha$,

$$\Rightarrow \operatorname{Prob}\left(\frac{\hat{s}(k) + u(k) - \hat{d}(k)}{V_{d,k}} \geq \frac{d(k) - \hat{d}(k)}{V_{d,k}}\right) \geq \alpha \tag{11.7}$$

with $\hat{d}_k$ = demand mean for period *k*. Note that $\left(\frac{d(k) - \hat{d}(k)}{V_{d,k}}\right)$, the Gaussian random variable with identical distribution to d_k

$$\Rightarrow \phi_{d,k}\left(\frac{\hat{s}(k) + u(k) - \hat{d}(k)}{V_{d,k}}\right) \geq \alpha \tag{11.8}$$

where $\phi_{d,k}$ is indefinitely derivable and strictly increasing.

Thus, (11.8) $\Rightarrow \frac{\hat{s}(k) + u(k) - \hat{d}(k)}{V_{d,k}} \geq \phi_{d,k}^{-1}(\alpha) \Leftrightarrow \hat{s}(k) + u(k) - \hat{d}(k) \geq V_{d,k} \times \phi_{d,k}^{-1}(\alpha)$

$\Leftrightarrow u(k) \geq V_{d,k} \times \phi_{d,k}^{-1}(\alpha) + \hat{d}(k) - \hat{s}(k)$.

Thus $\operatorname{Prob}\left(\hat{s}(k+1) \geq 0\right) \geq \alpha \Rightarrow \left(U_k \geq V_{d,k} \times \phi_{d,k}^{-1}(\alpha) + \hat{d}(k) - \hat{s}(k)\right)$.

The deterministic equivalent problem of the stochastic production/maintenance planning problem at hand should take into account the following features:

- the linearity of the economic balance equation;
- the convexity of the functional cost;
- the demand random variations, together described as Gaussian processes.

As a result, the equivalent deterministic model can be now formulated as follows:

$$Z(u) = C_s \times \left(\hat{s}(h)^2\right) + \sum_{k=0}^{h-1}\left[C_s \times \hat{s}(k)^2 + C_{pr} \times u(k)^2\right] + C_s \times (\sigma_{\mathrm{d}})^2 \times \frac{h\times(h-1)}{2},$$

subject to:

$$\hat{s}(k+1) = \hat{s}(k) + u(k) - d(k), \qquad k = 0,1,\ldots,h-1,$$

$$u(k) \ge U_\alpha(S_k, \alpha) \quad U_\alpha(S_k, \alpha) = V_{d,k} \times \phi^{-1}_{d,k} + \hat{d}(k) - \hat{S}(k), \qquad k = 0,1,\ldots,h-1,$$

$$0 \le u(k) \le U_{\max}, \qquad k = 0,1,\ldots,h-1.$$

11.2.2.3 Maintenance Policy

The Maintenance Cost Model in the Case of Minimal Repair Policy

We consider the well-known PM policy with minimal repair (Faulkner, 2005). The planning horizon [*0*, *H*], is divided equally into *N* parts of duration *T* each, with PM actions or replacements scheduled at times $k\times T$ (k=1, 2,…., *N*), with *NT=H*. Each replacement or PM action restores the unit to the as-good-as-new condition. When a unit fails between replacements or PM actions, only minimal repair is made, and hence the failure rate remains undisturbed. It is assumed that the repair and PM or replacement times are negligible.

Letting $\lambda(t)$ denote the machine failure rate function and

$L(T) = \int_0^T \lambda(t)dt$, the expected cost incurred over the interval [*0, T*] is:

$$\tilde{C}(1) = c_1 \times L(T) + c_2 = c_1 \times L\left(\frac{H}{N}\right) + c_2.$$

Thus, the total expected cost until time *H* is:

$$C(N) = N \times \tilde{C}(1) = N \times \left[c_1 \times L\left(\frac{H}{N}\right) + c_2\right]$$

$$\Rightarrow C(N) = N \times \left[c_1 \times \int_0^{H/N} \lambda(t)dt + c_2\right] \tag{11.9}$$

with $N \in N^*$.

Since the existence of an optimal PM interval *T** in the case of an increasing failure rate has been proven (Lyonnet, 2000), it follows that *N** exists.

Maintenance Policy Optimization

In this subsection, we seek to optimize the cost model associated with the PM with minimal repair policy derived above. Note that the production plan over the horizon *H* established in the previous subsection has an impact on the failure rate $\lambda(t)$. Consequently, the objective here is to take into account the production plan in determining the optimal number of partitions *N** of PM actions to be carried out, which in turn means that the PM action takes place at *T* =H/N* tu*.

11.2.3 Influence of Manufacturing System Deterioration on the Optimal Production Plan

11.2.3.1 Analytical Derivation

This section deals with the influence of the maintenance cost on the optimal production plan. Letting λ_k denote the failure rate at time *k* and augmenting Eq. 11.4 with a penalty term, which accounts for the deterioration of the system, the formulation becomes:

$$\min_u Z(u) = \min_u \begin{bmatrix} C_s \times \left(\hat{s}(h)^2\right) + \sum_{k=0}^{h-1} \left[C_s \times \hat{s}(k)^2 + C_{pr} \times u(k)^2 \right] \\ +C_s \times (\sigma_d)^2 \times \frac{h \times (h-1)}{2} + C_\lambda \times \sum_{k=0}^{h} (\lambda_k)^2 \end{bmatrix} \tag{11.10}$$

subject to:

$$\hat{s}(k+1) = \hat{s}(k) + u(k) - d(k), \qquad k = 0,1,..., h-1,$$

$$\text{Prob}\left[\hat{s}(k+1) \geq 0\right] \geq \alpha, \qquad k = 0,1,..., h-1,$$

$$0 \leq u(k) \leq U_{max}, \qquad k = 0,1,..., h-1.$$

For each period *k*, we employ the production rate $u(k)$ earlier established by the optimal production plan. Therefore the failure rate in each interval changes according to the interval's production rate and the failure rate at the beginning of the interval (Figure11.3). Formally, the failure rate in the interval *k* is expressed as follows, where $\lambda(k = 0) = \lambda_0$:

$$\lambda_k(t)=\sum_{j=1}^{k-1}\frac{u(j)}{U_{\max}}\times\left(\lambda_0\left(j\times\Delta t\right)-\lambda_0\left(\left(j-1\right)\times\Delta t\right)\right)$$
$$+\frac{u(k)}{U_{\max}}\times\left(\lambda_0\left(t\right)-\lambda_0\left(\left(k-1\right)\times\Delta t\right)\right)\qquad\forall t\in\left[\left(k-1\right)\times\Delta t,\ k\times\Delta t\right].\tag{11.11}$$

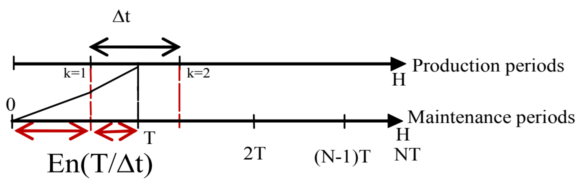


Fig. 11.3 Failure rate during *H*

11.2.3.2 Numerical Example

A company, whose sales are subject to seasonal fluctuating demand, tries to develop an aggregated production plan which minimizes total costs over a finite planning horizon: H = 18 months. The strategy discussed in the previous section has been employed here to provide a decision policy for managers.

The monthly mean demands $\left(\hat{d}_k\right)$ are as follows:

$\{d_1 = 431$, $d_2 = 444$, $d_3 = 442$, $d_4 = 340$, $d_5 = 392$, $d_6 = 400$, $d_7 = 350$, $d_8 = 400$, $d_9 = 350$, $d_{10} = 370$, $d_{11} = 395$, $d_{12} = 415$, $d_{13} = 431$, $d_{14} = 444$, $d_{15} = 442$, $d_{16} = 340$, $d_{17} = 392$, $d_{18} = 375\}$.

The other parameters are: C_{pr} =3 *mu*, C_s = 2 *mu/k*, $U_{\max}$ = 500 units/period, σ_d = 14, α = 0.95. We assume that the failure time of the machine is characterized by a Weibull distribution with scale and shape parameters respectively equal to β = 100 and α = 2 (consequently, the machine has a linear deterioration law). Thus the failure rate can be expressed as follows:

$$\lambda_i(t)=\lambda_{i-1}(\Delta t)+\frac{U(i)}{U_{\max}}\times\lambda(t)\tag{11.12}$$

where $\lambda(t)=\frac{\alpha}{\beta}\times\left(\frac{t}{\beta}\right)^{\alpha-1}$.

Applying the numerical procedure developed above, we have the optimal production plan for two cases, namely:

1. We do not consider a penalty on the deterioration (C_λ = 0).
2. We consider a penalty on the deterioration (C_λ = 3000).

The result is exhibited in Figure 11.4.

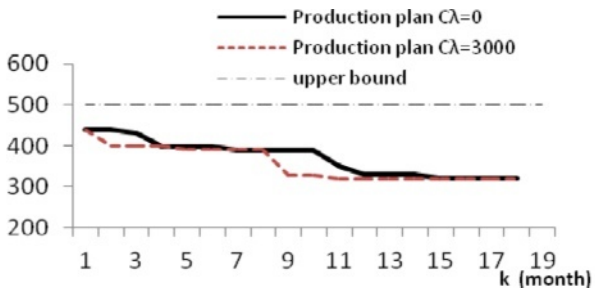


Fig. 11.4 Optimal production plan

We noted that the optimal production plan obtained in case 2 yields smaller production rates than does the production plan obtained in case 1.

11.2.4 Optimization of the Maintenance Policy

11.2.4.1 Analytical Study

In the previous section, we found that there exists a significant relationship between the production plan and the deterioration of the machine. That is why we will now focus on an intelligent joint optimization strategy in which we consider both the maintenance production elements. Firstly, we determine the optimal production plan minimizing the average inventory and production costs. Secondly, by using this optimal production plan in the maintenance cost formulation, we will determine the optimal maintenance strategy characterized by the number N^* of PM actions to perform. The analytical model, which calculates the total expected cost of maintenance, is developed under the following assumptions.

- Probability density functions of time to failure and time to perform maintenance activities are known.
- Costs c_2 and c_1 incurred by preventive and corrective maintenance actions are known and constant, with $c_1 >> c_2$.
- PM is performed as soon as the age of the machine reaches T.
- Failures are detected instantaneously.

The analytic expression of the total expected maintenance cost is as follows, with $T = H/N$, $N \in \mathbb{N}^*$:

$$C(N) = N \times \left[c_1 \times L(T) + c_2 \right].$$

Recall that $L(T) = \int_0^T \lambda_s(t)dt$

where $\lambda_s(t)$ represents the failure rate of the machine in $[0\,,\,T]$. Also recall that the failure rate changes for each interval $[i\Delta t,\ (i+1)\Delta t]$ according to the optimal production plan obtained. Therefore, the $L(T)$ expression can be derived as follows:

$$L(T) = \sum_{i=1}^{Int\left(\frac{T}{\Delta t}\right)} \int_0^{\Delta t} \lambda_i(t)dt + \int_{Int\left(\frac{T}{\Delta t}\right)\times\Delta t}^{T} \frac{u_{Int\left(\frac{T}{\Delta t}\right)+1}}{U\max} \times \lambda(t)dt \quad (11.13)$$

Recall that we have formulated $\lambda_i(t)$ Eq. 11.11 as follows:

$\lambda(t) = \frac{\alpha}{\beta}\times\left(\frac{t}{\beta}\right)^{\alpha-1}$ where $\lambda_i(t) = \lambda_{i-1}(\Delta t) + \frac{U(i)}{U_{\max}}\times\lambda(t)$.

Hence we can express $\lambda_i(t)$ as

$$\lambda_i(t) = \lambda_0(t_0) + B_i + \frac{U(i)}{U_{\max}}\times\lambda(t)$$

where $B_i = \sum_{j=1}^{i-1} \frac{U(j)}{U_{\max}}\times\lambda(\Delta t)$ and $B_1 = 0$, $\lambda_0(t_0) = \lambda_0$.

Replacing the expression of $\lambda_i(t)$ in Eq. 11.11, we have:

$$L(T) = \sum_{i=1}^{Int\left(\frac{T}{\Delta t}\right)} \int_0^{\Delta t} \lambda_0(t_0) + B_i + \frac{u(i)}{U\max}\times\lambda(t)dt + \frac{u_{Int\left(\frac{T}{\Delta t}\right)+1}}{U\max}\times \int_{Int\left(\frac{T}{\Delta t}\right)\times\Delta t}^{T} \lambda(t)dt$$

$$L(T) = Int\left(\frac{T}{\Delta t}\right)\times\Delta t\times\lambda_0 + \sum_{i=1}^{Int\left(\frac{T}{\Delta t}\right)} \int_0^{\Delta t} B_i dt + \sum_{i=1}^{Int\left(\frac{T}{\Delta t}\right)} \int_0^{\Delta t} \frac{u(i)}{U\,max}\times\lambda(t)dt + \frac{u_{Int\left(\frac{T}{\Delta t}\right)+1}}{U\,max}\times \int_{Int\left(\frac{T}{\Delta t}\right)*\Delta t}^{T} \lambda(t)dt. \quad (11.14)$$

Finally, the problem reduces to finding an optimal number of partitions N^* which minimizes $C(N)$, with

$$C(N) = N\times\left[c_1\times L(T) + c_2\right].$$

Lemma 11.2: *∃ N* minimizing C(N) if* $\theta_N \leq c_2/c_1 \leq \theta_{N-1}$ *With* $\theta_N = N \times L(H/N) - (N+1) \times L(H/N+1)$.

Proof: It is easy to see that

$$\lim_{N\to 1} C(N) = \tilde{C}(1) = c_1 \times L\left(\frac{H}{1}\right) + c_2$$

$$\lim_{N\to\infty} C(N) \to \infty . \tag{11.15}$$

Therefore we have:

- $C(N+1) - C(N) = (N+1) \times \left[c_1 \times L\left(\frac{H}{N+1}\right) + c_2\right] - N \times \left[c_1 \times L\left(\frac{H}{N}\right) + c_2\right]$

1. $C(N+1) - C(N) \geq 0$;
2. $(N+1) \times \left[c_1 \times L\left(\frac{H}{N+1}\right) + c_2\right] - N \times \left[c_1 \times L\left(\frac{H}{N}\right) + c_2\right] \geq 0$;
3. $N \times L\left(\frac{H}{N}\right) - (N+1) \times L\left(\frac{H}{N+1}\right) \leq \frac{c_2}{c_1}$;
4. $\theta_N \leq \frac{c_2}{c_1}$. (11.16)

with $\theta_N = N \times L\left(\frac{H}{N}\right) - (N+1) \times L\left(\frac{H}{N+1}\right)$.

Likewise,

- $C(N) - C(N-1) = N \times \left[c_1 \times L\left(\frac{H}{N}\right) + c_2\right] - (N-1) \times \left[c_1 \times L\left(\frac{H}{N-1}\right) + c_2\right]$

1. $C(N) - C(N-1) \leq 0$;
2. $N \times \left[c_1 \times L\left(\frac{H}{N}\right) + c_2\right] - (N-1) \times \left[c_1 \times L\left(\frac{H}{N-1}\right) + c_2\right] \leq 0$;
3. $(N-1) \times L\left(\frac{H}{N-1}\right) - N \times L\left(\frac{H}{N}\right) \geq \frac{c_2}{c_1}$;
4. $\theta_{N-1} \geq \frac{c_2}{c_1}$. (11.17)

with $\theta_{N-1} = (N-1) \times L\left(\frac{H}{N-1}\right) - N \times L\left(\frac{H}{N}\right)$.

Formulas 11.15, 11.16 and 11.17 show the existence of *N** minimizing *C*(*N*).

11.2.4.2 Numerical Example

Let us take the same data as in Sect. 11.2.3. Applying the numerical procedure described in Sect. 11.2.2.2, we obtain the optimal production plan exhibited in Table 11.1.

Table 11.1 Optimal production plan

k	1	2	3	4	5	6	7	8
D	431	444	442	340	392	375	392	400
U^*	500	500	500	342.57	496.85	340.72	426.27	416.13
k	9	10	11	12	13	14	15	16
D	350	370	395	415	431	444	442	340
U^*	249.18	410.32	445.41	455.32	463.26	470.21	437.96	134.32
k	17	18	19	20	21	22	23	24
D	392	375	400	420	350	340	392	370
U^*	496.85	340.72	450.41	460.32	208.85	319.83	496.85	325.63

We next use the above optimal production plan in the total expected maintenance cost expression, $C(N)$, formulated in Sect. 11.2.4.1. Concerning the numerical example, assume the same failure time distribution as in Sect. 11.2.3.2. Moreover, in order to simplify the numerical evaluation, we substitute the expression of $\lambda(t)$ into the RHS of $L(T)$. Further development of the latter equation yields:

$$L(T)=\Delta t\left[\operatorname{Int}\left(\frac{T}{\Delta t}\right)\cdot\lambda_0\left(t_0\right)+\frac{(\Delta t)^{\alpha-1}}{\beta^{\alpha}U_{\max}}\sum_{i=1}^{\operatorname{Int}\left(\frac{T}{\Delta t}\right)}U_i+\frac{\alpha}{\beta^{\alpha}U_{\max}}\sum_{i=1}^{\operatorname{Int}\left(\frac{T}{\Delta t}\right)}\sum_{j=1}^{i-1}U_j\cdot\left(\Delta t\right)^{\alpha-1}\right]$$
$$+\frac{U_{\operatorname{Int}\left(\frac{T}{\Delta t}\right)+1}}{U_{\max}}\cdot\frac{1}{\beta^{\alpha}}\cdot\left[T^{\alpha}-\operatorname{Int}\left(\frac{T}{\Delta t}\right)^{\alpha}\right].$$

Finally, the maintenance optimization problem is described as follows, where we seek to find the minimizing value of N:

$$C(N)=N\cdot\left[c_1\cdot\left[\begin{array}{l}\Delta t\left[\operatorname{Int}\left(\frac{T}{\Delta t}\right)\cdot\lambda_0\left(t_0\right)+\frac{(\Delta t)^{\alpha-1}}{\beta^{\alpha}U_{\max}}\sum\limits_{i=1}^{\operatorname{Int}\left(\frac{T}{\Delta t}\right)}U_i+\frac{\alpha}{\beta^{\alpha}U_{\max}}\sum\limits_{i=1}^{\operatorname{Int}\left(\frac{T}{\Delta t}\right)}\sum\limits_{j=1}^{i-1}U_j\cdot\left(\Delta t\right)^{\alpha-1}\right]\\ +\frac{U_{\operatorname{Int}\left(\frac{T}{\Delta t}\right)+1}}{U_{\max}}\cdot\frac{1}{\beta^{\alpha}}\cdot\left[T^{\alpha}-\operatorname{Int}\left(\frac{T}{\Delta t}\right)^{\alpha}\right]\end{array}\right]+c_2\right].$$

Assume that Δt= 1 *tu*. Figure 11.5 presents the histogram of the total expected maintenance cost, $C(N)$, as a function of N.

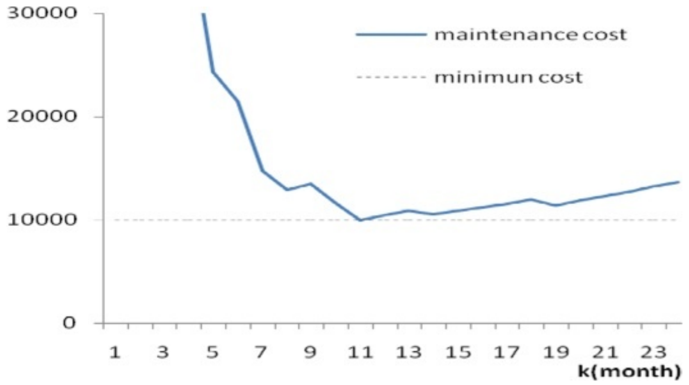


Fig. 11.5 Curve of total cost expected of maintenance according to *N*

We conclude that the optimal number of PM actions is N^* = 11. This means that over the finite horizon *H* of 56 months, 11 PM actions must be carried out at intervals of $T^* = H/N^*$ =2.18 *tu*. The corresponding minimal total expected maintenance cost is C^* = 10,007.0054 *mu*.

11.3 Intelligent Periodic Preventive Maintenance Policy in Finite Horizon with an Adaptive Failure Law

11.3.1 Problem Description

This study considers a navy ship (the system) that must carry out a set of ordered missions. Each mission i is defined by a duration δ_i, and deteriorates the vessel in accordance with operational and environmental conditions, respectively defined by $z_{i,1}$ and $z_{i,2}$. The deterioration is represented by an increase in the nominal hazard rate, the latter of which represents the instantaneous failure risk. The Cox model (Cox, 1972) is used since it establishes a parametric relationship between risk factors (related to the operational and environmental conditions of each mission) and the uptime distribution. The method relies mainly on the assumption of proportional hazards, which assumes that each factor affects the life steadily over time.

Let:

- $\lambda_0(t)$ = hazard rate under nominal conditions;

- $\lambda(t, Z_i)$ = hazard rate at time t under conditions Z_i (vector); and
- $g(Z_i) = e^{\sum_{j=1}^{2} b_j \cdot z_{i,j}}$, a risk function where b_i represents the weighting of Z_i.

For a mission i, the Cox model is given by:

$$\lambda(t, Z_i) = \lambda_0(t) \times g(Z_i). \tag{11.18}$$

It is assumed that the business plan, i.e. the ordered set of all missions to be performed, has already been specified. Thus, we focus only on the intelligent determination of the optimal maintenance plan, which minimizes maintenance costs. When the ship fails, a minimal corrective maintenance (repair) activity is performed, the immediate goal being to get the ship running again. Such a minimal repair, characterized by a constant cost and a fixed duration, does not have any influence on the system hazard rate. To reduce maintenance costs, it is necessary to decrease the average number of failures that may arise when the business plan is performed. We have chosen to carry out periodic PM. The PM actions are imperfect, as they will be carried out at sea, i.e. under less than ideal conditions. This effect is modelled by a reduction of the hazard rate following the age-based PM model. When the maintenance quality index takes the value 0, maintenance is considered as minimal, whereas maintenance is considered as perfect when the value of the index is equal to 1. The cost of PM consists of a fixed cost and a variable cost, which depends on the quality of maintenance performed. The same principle is used to determine the time required to accomplish the maintenance activity. At the end of the business plan, a renewal is performed to restore the ship to a state as-good-as-new. In order to develop an analytical model of the problem, we introduce the following notation:

$\Gamma(N, \rho)$	total cost of maintenance
C_{R}	cost of renewal
C_{C}	cost of minimal repair
$C_{\mathrm{P}}^{\mathrm{f}}$	fixed cost of imperfect maintenance
$C_{\mathrm{P}}^{\mathrm{v}}$	variable cost of imperfect maintenance
ρ	maintenance quality index
μ_{C}	time to perform minimal repair
$\mu_{\mathrm{P}}^{\mathrm{f}}$	fixed time to perform imperfect maintenance
$\mu_{\mathrm{P}}^{\mathrm{v}}$	variable time to perform imperfect maintenance
$\phi\left(\sum_{m=1}^{M} \delta_m, N, \rho\right)$	average number of failures
N	number of PM actions

u_m	date at which the reliability associated with the beginning of mission m is equal to the reliability at the end of the previous mission
$R_m(u_m)$	reliability of mission m at time u_m
$b(i)$	mission performed at the beginning of interval i
$e(i)$	mission performed at the end of the interval i
Δ_m	duration of a PM activity

11.3.2 Model Formulation and Intelligent Determination of the Optimal Solution

The total cost of maintenance is given by:

$$\gamma(N,\rho)=C_{\mathrm{C}}\times\phi\left(\sum_{m=1}^{M}\delta_m,N,\rho\right)+\left(C_{\mathrm{P}}^{\mathrm{f}}+C_{\mathrm{P}}^{\mathrm{v}}\wedge\rho\right)\times N+C_{\mathrm{R}}. \tag{11.19}$$

As operational and environmental conditions depend on the missions to be achieved, the average number of failures occurring during the business plan is given by:

$$\phi\left(\sum_{m=1}^{M}\delta_m,\,N,\,\rho\right)=\sum_{i=1}^{N}\sum_{m=b(i)}^{e(i)}\int_{u_m}^{u_m+\Delta_m}\lambda\left(t,\,Z_m\right). \tag{11.20}$$

If missions m and $m-1$ are executed, u_m can be determined by:

$$R_m\left(u_m\right)=R_{m-1}\left(u_{m-1}+\delta_{m-1}\right). \tag{11.21}$$

If m corresponds to the first mission of the interval i, the computation of u_m varies because a PM was performed. With the age-based model, the virtual age is given by:

$$u_m=\left(1-\rho\right)\times R_m^{-1}\left(R_{m-1}\left(u_{m-1}+\delta_{m-1}\right)\right). \tag{11.22}$$

The shaded area in Figure 11.6 represents the difference in average number of failures between two missions with different risk factors. As the conditions are worse for the second mission, the average number of failures is higher.

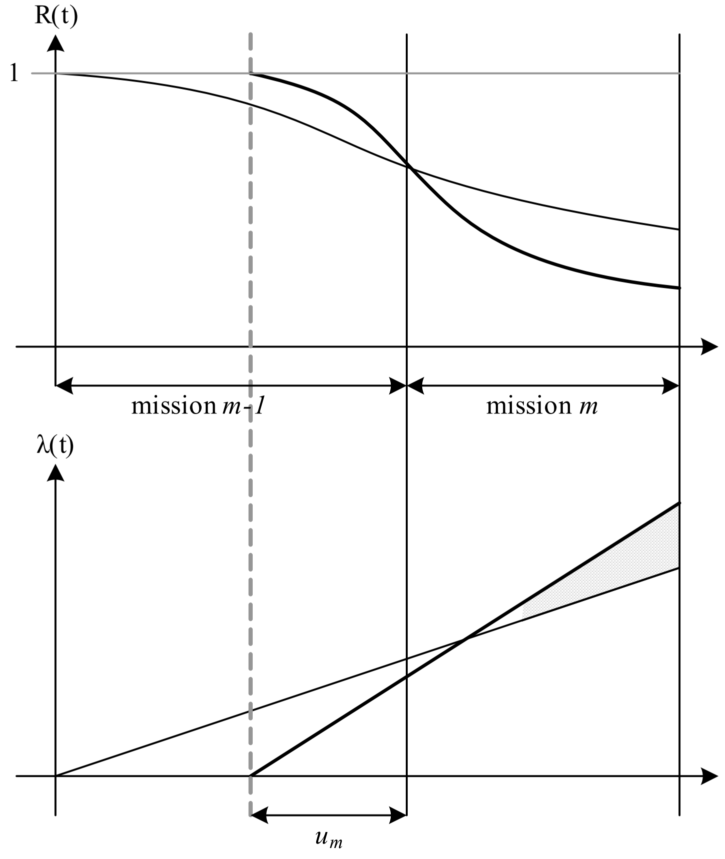


Fig. 11.6 Schematic presentation of u_m for two consecutive missions

For a Weibull failure law associated with the Cox model, u_m is defined by:

$$u_m = \left(\frac{g\left(Z_{m-1}\right)}{g\left(Z_m\right)} \right)^{1/\beta} \times \left(u_{m-1} + \delta_{m-1}\right) \quad \forall m > 1, \tag{11.23}$$

$u_1 = 0$.

The maintenance plan time is divided into $\left(N+1\right)$ regular intervals. Each interval has a constant duration equal to $\left(\sum_{m=1}^{M} \delta_m \right) \Big/ \left(N+1\right)$.

The PM activities can be carried out during the missions. Hence, it is necessary to determine the missions that begin and finish each interval. The four possible cases are given by

$$\Delta_m = \begin{cases} \delta_m & \text{if } m \neq b(i) \text{ and } m \neq e(i), \\ \delta_{e(i)} - \left(\sum_{j=1}^{e(i)} \delta_j - \frac{\sum_{m=1}^{M} \delta_m}{N+1} \times i \right) & \text{if } m \neq b(i) \text{ and } m = e(i), \\ \sum_{j=1}^{b(i)} \delta_j - \frac{\sum_{m=1}^{M} \delta_m}{N+1} \times (i-1) & \text{if } m = b(i) \text{ and } m \neq e(i), \\ \frac{\sum_{m=1}^{M} \delta_m}{N+1} & \text{if } m = b(i) \text{ and } m = e(i). \end{cases} \tag{11.24}$$

For this maintenance policy, two variables must be optimized: the number of PM activities N and the maintenance quality index ρ. The solution is obtained by solving the following equations:

$$\frac{\partial C(N,\rho)}{\partial N} = 0 \text{ and } \frac{\partial C(N,\rho)}{\partial \rho} = 0. \tag{11.25}$$

To solve this problem, the numerical procedure described in Figure 11.7 is used.

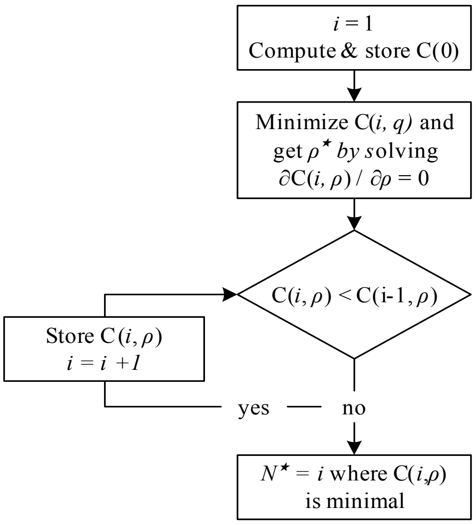

$i = 1$
Compute & store C(0)
Minimize C(i, q) and get $\rho^\star$ by solving ∂C(i, ρ) / $\partial\rho$ = 0
C(i, ρ) < C(i-1, ρ)
Store C(i, ρ)
$i = i + 1$
yes
no
$N^\star = i$ where C(i,ρ) is minimal

Fig. 11.7 Numerical procedure to determine $N^\star$ and $\rho^\star$

In the first step of this numerical procedure, the cost of maintenance, without PM, is computed. The maintenance quality index need not be specified here since no PM is performed, i.e., $\rho = 0$. Now, the availability of the system, denoted $A(N,\ \rho)$, is given by:

$$A(N,\ \rho) = \frac{\sum_{m=1}^{M} \delta_m}{\sum_{m=1}^{M} \delta_m + \mu_C \times \phi\left(\sum_{m=1}^{M} \delta_m,\ N,\ \rho\right) + \left(\mu_P^f + \mu_P^v \wedge \rho\right) \times N}. \tag{11.26}$$

If we wish to achieve a minimum availability, it is necessary to amend the second step of the solution procedure as follows: "Minimize $\Gamma(N,\ \rho)$ and get $\rho^\star$ by solving $\partial\Gamma(N,\ \rho)/\partial\rho = 0$ with $A(N,\ \rho)$ > minimal availability". To illustrate this methodology, a numerical example is presented in the next subsection.

11.3.3 Numerical Example

The different parameters are given below

- Weibull failure law associated with the Cox model:
  - β (shape parameter) = 2
  - η (scale parameter) = 600
  - b_1 (weight of operational condition) = 1.4
  - b_2 (weight of environmental condition) = 0.5
- C_R = 3000 *mu*
- C_C = 1500 *mu*
- C_P^f = 200 *mu*
- C_P^v = 1000 *mu*
- μ_C = 12 *tu*
- $\mu_\mathrm{P}^\mathrm{f}$ = 2 *tu*
- $\mu_\mathrm{P}^\mathrm{v}$ = 4 *tu*

Without applying the PM policy, the cost of maintenance (consisting of failures only) amounts to 13,635.10 *mu*, the time required for corrective maintenance activities is 85.08 *tu*, and the availability of the system is 89.81%. These results will be used to estimate the gains achieved by the periodic PM. (see Table 11.2).

Table 11.2. List of performed missions

Duration δ_i	Operational conditions $z_{i,1}$	Environmental conditions $z_{i,2}$
121	1	2
72	0	0
113	1	0
86	2	0
82	1	1
107	0	2
74	1	1
95	2	2

By applying the numerical procedure of Sect. 11.3.2, the optimal PM quality index was determined for each number of PM actions. These results are given in Table 11.3.

Table 11.3. Intelligent determination of the optimal quality maintenance for fixed number PM actions

N	$\rho^\star$	$\Gamma\left(N,\, \rho^\star\right)$	$A\left(N,\, \rho^\star\right)$
1	0.87	11830.02	91.34
2	0.82	10803.014	92.04
3	0.78	10138.43	92.38
4	0.75	9945.62	92.27
5	0.72	9624.03	92.28
6	0.69	9545.64	92.05
7	0.67	9520.82	91.8
8	0.65	9381.82	91.66
9	0.64	9336.20	91.45
10	0.62	9341.31	91.18
11	0.61	9396.87	90.88

We can state from Table 11.3 and Figure 11.8 that the minimum cost is obtained for nine PM activities. Given the optimal number of PM actions, Figure 11.8 illustrates the variation of maintenance costs as a function of the maintenance quality index.

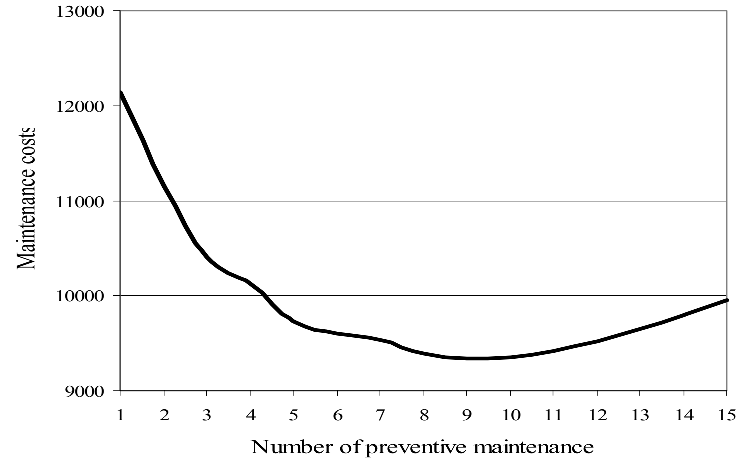


Fig. 11.8 Evolution of maintenance costs depending on the number of PM actions

For the optimal values of the variables (N^* and ρ^*), the economic gain, compared to a maintenance policy consisting only of minimal repairs, is 4298.90 m_u. The availability of the system has also been improved by 1.64%. Moreover, if a minimum availability constraint is taken into consideration, it is necessary to determine a new pair (N^* and ρ^*). For instance, Figure 11.9 and Table 11.4 give the new results with the added constraint $A\left(N,\ \rho^*\right) > 92.5\%$.

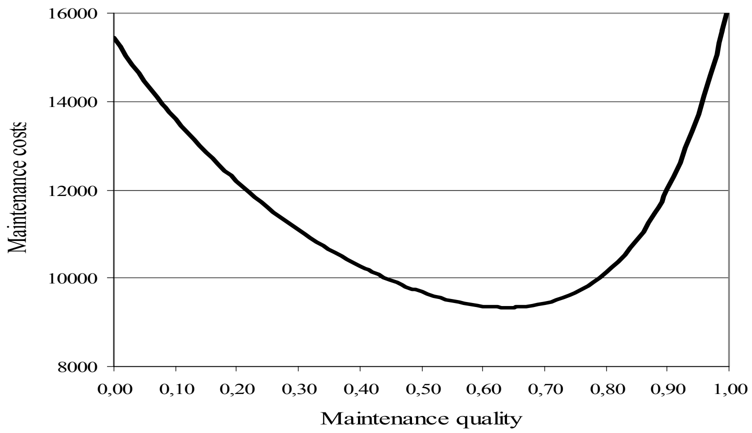


Fig. 11.9 Evolution of maintenance costs depending on the action quality

Table 11.4 Intelligent determination of the optimal maintenance quality index given *N* and subject to an availability constraint

N	$\rho^\star$	$C(N, \rho^\star)$	$D(N, \rho^\star)$
1	–	–	–
2	–	–	–
3	0.83	10182.84	92.87
4	0.86	10254.98	92.71
5	0.85	10081.63	92.61
6	–	–	–

Note that the additional availability constraint restricts the feasible number of PM actions to between three and five, while the optimal maintenance quality index is higher than in the previous unconstrained case. Hence, the total cost of maintenance is higher.

11.4 Conclusions

In this chapter, we presented two studies investigating new intelligent integrated maintenance and production or service strategies, which deal with complex reliability problems. The first study initially described a sequential constrained linear-quadratic stochastic production-planning problem in which a random demand must be satisfied and the machine is subject to random failure. A minimal repair is performed at every failure, with PM actions scheduled according to the manufacturing system history so as to reduce the failure frequency. The influence of the deterioration of the manufacturing system on the optimal production plan was shown. Next, in view of the influence of the production rate on the machine failure rate, we used the optimal production plan in the maintenance problem formulation, in order to establish a maintenance schedule.

In the second study, we proposed a failure law that can vary over time, not only with respect to the PM actions, but also as a function of changes in operating and/or environmental conditions. The change in operating and/or environmental conditions is modelled by a multiplication of the failure rate by a factor which represents their effect. In addition, PM actions are imperfect and their cost and duration depend on a maintenance quality index. The goal of the study is to determine intelligently the number of PM actions, which must be performed in order to minimize the cost under a threshold availability constraint. Naturally, a complementary approach is also possible, i.e. maximize the availability subject to maximum maintenance costs.

References

Barlow R, Hunter L (1960) Optimum preventive maintenance policies. Oper. Res., 8(1):90–100

Brown M, Proschan F (1983) Imperfect repair. J. of Appl. Prob. 20(4):851–859

Buzacott JA, Shanthikumar JG (1993) Stochastic models of manufacturing systems. Prentice Hall, Englewood Cliffs, NJ

Cheung KL, Hausmann WH (1997) Joint optimization of preventive maintenance and safety stock in an unreliable production environment. Naval Res. Logist. (Int. J.) 44:257–272

Cox DR (1972) Regression models and life tables (with discussion). J. Royal Stat. Soci., Series B 34:187–220

Dedopoulos I, Smeers Y (1998) An age reduction approach for finite horizon optimization of preventive maintenance. Comput. Ind. Eng. 34(3):643–654

Faulkner LL (2005) Maintenance, replacement and reliability theory and applications. Taylor and Francis, USA

Dellagi S, Rezg N, Xie X (2007) Preventive maintenance of manufacturing systems under environmental constraints. Int. J. Prod. Res., 45 (5):1233–1254

Hajej Z, Dellagi S, Rezg N (2009) Optimization of a maintenance strategy with considering the influence of the production plan on the manufacturing system degradation. In: Proceedings of the 13th IFAC Symposium on Information Control Problems, Moscow

IMS: Center for Intelligent Maintenance Systems. http://www.imscenter.net/

Lie CH, Chun YH (1986) An algorithm for preventive maintenance policy. IEEE Trans. Reliability. 35:71–75

Malik MAK (1979) Reliable preventive maintenance scheduling. IIE Trans. 11(3):221–228

Martorell S, Sanchez A, Serradell V (1999) Age-dependent reliability model considering effects of maintenance and working conditions. Reliab. Eng. Syst. Safety, 64(1):19–31

Özekici S (1995) Optimal maintenance policies in random environments. Eur. J. Oper. Res., 82 (2):283-294

Lyonnet P (2000) La maintenance : mathématiques et méthodes. Editions TEC and DOC, Paris

Rezg N, Xie X, Mati Y (2004) Joint optimization of preventive maintenance and inventory control in a production line using simulation. Int. J. Prod. Res., 44:2029–2046

Rezg N, Dellagi S, Chelbi A (2008) Optimal strategy of inventory control and preventive maintenance. Int. J. Prod. Res., 46(19):5349–5365

Silva F, Cezarino W (2004) An optimal production policy applied to a flow-shop manufacturing system. Brazilian J. Oper. Prod. Manag. 1(1):ISSN 1679–8171

Valdez-Flores C, Feldman RM (1989) A survey of preventive maintenance models for stochastically deteriorating Single-unit systems. Naval Res. Logist. (Int. J.) 36:419–446

Wang H, Pham H (1996) Optimal maintenance policies for several imperfect maintenance models. Int. J. Syst. Sci. 27(6):543–550

Chapter 12
Enhancing the Effectiveness of Multi-pass Scheduling Through Optimization via Simulation

T. Yoo[1], H.-B. Cho[2] and E. Yücesan[3]

Abstract While difficult in their own right, scheduling problems are further complicated by the concurrent flow of various parts, the sharing of different types of resources, and the random occurrence of disruptive events. To deal with such complexity, multi-pass scheduling has been developed. Successful application of multi-pass scheduling, however, largely depends on its ability to quickly and effectively select the best decision-making rule. The objective of the present work is to enhance the performance of multi-pass scheduling through optimization via simulation. To this end, we combine random search and statistical selection to create a potent approach for optimization over a large but finite solution space when the objective function must be evaluated using noisy estimation. The nested partitions method is a global search strategy that is continuously adapted via a partitioning of the feasible solution region. By allocating computing resources to potentially critical design points, the optimal computing budget allocation method, in turn, provides an adaptive sampling mechanism from a feasible region. Through carefully designed experiments, the proposed approach is shown to perform markedly better than naive multi-pass scheduling.

Keywords Multi-pass scheduling, scheduling rule, real-time scheduling, nested partitions, optimal computing budget allocation

[1] T. Yoo
Samsung Electronics CO., LTD., Republic of Korea

[2] H.-B. Cho
Department of Industrial Engineering, Pohang University of Science and Technology, San 31 Hyoja, Pohang 790-784, Republic of Korea

[3] E. Yücesan (✉)
Technology and Operations Management Area, INSEAD, Boulevard de Constance 77305, Fontainebleau Cedex, France
e-mail: Enver.Yucesan@insead.edu

12.1 Introduction

Scheduling is a field that is abundant with difficult problem instances for stochastic optimization. Traditionally, scheduling problems have been formulated using analytical methods such as mathematical programming or graph theory to seek optimal solutions; however, these methods involve simplifying assumptions that do not reflect the actual shop-floor status (Chunda and Mize, 1994). Combinatorial approaches such as genetic algorithms, simulated annealing, and tabu search have also been employed to obtain near-optimal solutions with limited success. Unfortunately, none of these techniques is appropriate for real-time scheduling because they cannot handle key operating characteristics of dynamic shop-floors such as random disturbances (e.g., machine breakdowns, tool breakage, and rush orders), concurrent flows of various parts, and sharing of different types of resources (Davis and Jones, 1988).

The difficulties associated with determining analytical solutions for real-time scheduling led to research on rule-based approaches, which typically employ various rules to prioritize jobs competing for the use of a given resource (e.g., a machine). A rule-based approach is associated with an event-driven job scheduling mechanism, in which each event requiring decision-making is resolved through a specific decision-making rule. However, past research indicates that the performance of a rule depends on the shop conditions at a particular instant (Jeong and Kim, 1998); it would therefore be preferable to change strategies dynamically and at the right instant in accordance with the current shop-floor conditions, instead of using a fixed scheduling rule for every scheduling period (Kutanoğlu and Sabuncuoğlu, 2002). This need for adaptability led to the development of multi-pass scheduling.

Multi-pass scheduling looks ahead into the future of the shop-floor through discrete-event simulation before the schedules are actually executed (Wu and Wysk, 1988, Cho and Wysk, 1993). For example, consider, as an arbitrary event, "a part finishes processing on a machine". The simulator can apply a particular rule in order to determine the next destination of the part in the presence of alternative routings. Such events may occur repeatedly as the simulation time advances. As a result, multi-pass scheduling would evaluate all possible rules for such future events using simulation, and then choose the best rule that can be executed in the actual shop-floor operations. Because this approach is simulation based, there is no need to make any unnecessarily restrictive assumptions about the shop-floor status. The simulation will be invoked whenever the rule combination needs updating. Furthermore, the multi-pass framework can be used not only to find the new best rule for every event requiring a decision, but also to reassess the future (predicted) performance. The multi-pass scheduling framework is depicted in Figure 12.1.

Several approaches have been developed to reduce the number of decision rules for each problem type in multi-pass scheduling, including expert systems (Wu and Wysk, 1988), neural networks (Cho and Wysk, 1993), hybrid approaches combining neural networks and genetic algorithms (Jones *et al.*, 1995), genetic algorithms (Yang and Wu, 2003), fuzzy algorithms (Sudiarso and Labib, 2002), and heuristic methods (Tormos and Lova, 2003). Whenever rules that are in use need to be updated, these approaches are employed to quickly recommend a small number of candidate rules. A more appropriate approach would consider current system conditions that influence the effectiveness of the decision rule. Thus, finding the best rule combination based on a particular shop condition is equivalent to the classification problem that maps each shop condition into a rule combination. It would therefore be ideal to find a perfect classification model that can map every possible shop-floor condition into a corresponding rule combination.

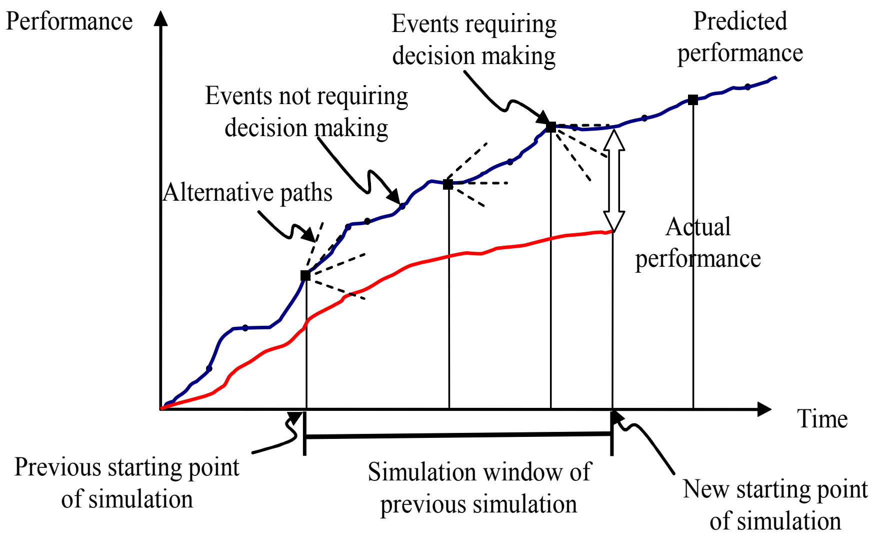

Performance
Events requiring decision making
Predicted performance
Events not requiring decision making
Alternative paths
Actual performance
Time
Previous starting point of simulation
Simulation window of previous simulation
New starting point of simulation

Fig. 12.1 Framework of the multi-pass scheduling approach

This idea has been deployed in an expert system prototype using a three-level top-down approach for rule selection. The expert system consists of a knowledge base, an inference engine, and a learning module to recommend a set of rules for a particular shop condition. The three environment parameters – system status, machine status, and job status – are identified; based on the values assumed by these parameters, the most important performance criterion is determined and the best rule is selected based on this criterion. Another approach is to combine a rule-based expert system, which is used to select several scheduling rules in a real-time environment, and a multi-pass simulator, which is used to determine the best scheduling rule (Wu and Wysk, 1988).

Neural networks have also been used to recommend candidate rules for multi-pass scheduling (Cho and Wysk, 1993). Five types of scheduling problems were identified in the context of an automated workstation. Whenever the rule being used needs to be updated, the neural network generates candidate rules for each problem type to be evaluated by the simulator. The neural network has seven input nodes, corresponding to the system status, and nine output nodes, one for each of the rules considered. Using the two best rules provided by the neural network, the simulator then selects the better one as a function of the status of the manufacturing system.

Inductive learning and genetic algorithms have also been used to construct relationships between shop conditions and rules (Chiu and Yih, 1995). Under this approach, the best rule is chosen at each decision point by using a genetic algorithm; the selected rule then forms a training case along with the shop conditions. Furthermore, the learning algorithm can modify the decision tree when new training cases are presented only if the change is significant. In addition, modified techniques based on neural networks and production rules (Kim *et al.*, 1998) have been used for identifying the relationships between shop conditions and rules.

Searching for the best alternative among a finite set of competing alternatives is a frequently encountered problem in many industrial applications such as job sequencing, scheduling, and resource allocation. Such problems are combinatorial in nature whereby the size of the feasible region grows exponentially in the input parameters, often leading to NP-hard optimization problems. For stochastic systems, such as job shops, the difficulty is exacerbated by the added randomness in job arrivals, concurrent flows, resource contention, and disruptions. More formally, the stochastic optimization problem can be formulated as:

$$\min_{\theta \in \Theta} f(\theta) \tag{12.1}$$

where Θ is a finite feasible region and $f: \Theta \to R$ is a performance function that may be subject to noise, ω. Since for any feasible point $\theta \in \Theta$, $f(\theta)$ cannot be evaluated analytically, simulation is often the only available tool for performance evaluation. However, as it is impractical to solve Eq. 12.1 for every sample realization, ω, we instead resort to solving the optimization problem:

$$\min_{\theta \in \Theta} E\left[f(\theta;\omega)\right]. \tag{12.2}$$

Simulation-based optimization is an active area of research (Andradóttir, 1998; Fu, 2002). When the number of alternatives is relatively small, ranking and selection, and multiple comparison methods can be successfully applied (Swisher *et al.*, 2003). These methods evaluate the performance of each alternative and identify the best alternative with a given probability of correct selection. When the number of alternatives grows larger, these methods become computationally prohibitive, necessitating random search techniques that would only consider a fraction of all the alternatives. Such techniques include simulated annealing (Alrefaei and Andradóttir, 1999), the stochastic ruler method (Alrefaei and Andradóttir, 2001), the stochastic comparison method (Gong *et al.*, 1999), the cross entropy method (Rubinstein 1999), the nested partitions (NP) method (Shi and Ólafsson, 2000), and COMPASS (Hong and Nelson, 2006). These techniques may move from a current best solution to a better solution, typically selected from among the neighbours of the current solution. Due to inherent randomness, the selection among neighbours has to be performed with care.

To solve the problem of optimization via simulation (OvS) as defined in (12. 2), we combine global random search using NP with statistical selection using optimal computing budget allocation (OCBA). Such a combination enables effective exploration of the solution space (i.e., the set of competing alternatives) and efficient exploitation of a given neighbourhood (i.e., optimal allocation of sampling effort). More specifically, we deploy the NP+OCBA methodology to enhance the efficiency of multi-pass scheduling for real-time shop-floor control. A key requirement of multi-pass scheduling for real-time shop-floor control is the ability to evaluate the rule combinations rapidly and effectively. The computational time depends on (i) the number of rule combinations to be evaluated and (ii) the number of simulation replications required for accurately estimating the performance of each rule combination. Under the NP+OCBA combination, the number of evaluated rule combinations is reduced by using the NP method while the number of simulation replications required for performance estimation is reduced by using the OCBA method. We demonstrate, through a designed experiment, the efficiency and effectiveness of the proposed approach, and compare it to other techniques frequently used in practice. Furthermore, our study quantifies the efficiency improvement in OvS obtained by embedding OCBA in NP within an illustrative case study on multi-pass scheduling.

Simulation has the unparalleled advantage of capturing arbitrary levels of system complexity. OvS, however, is computationally intensive. The method we deploy therefore combines an efficient random search method with statistical selection for guiding the search. While there exists a rich literature on OvS, our approach is closer to the work by Ólafsson (2004) and Pichitlamken *et al.* (2006). Ólafsson used Rinott's two-stage selection procedure for identifying the most promising region to explore by NP. As highlighted by Ólafsson, Rinott's approach is not only quite conservative as it is based on the least favourable configuration assumption, but may also be inefficient as it is driven by estimated variance, ignoring the mean performance – potentially spending significant computational effort on inferior alternatives with large variances. We overcome these limitations by using OCBA, which sequentially allocates further computing budget based on updated estimates of mean performance and its variance. Similarly, Pichitlamken *et al.* (2006) developed a sequential selection with memory approach in sampling from the most promising region. In our numerical tests, our approach compares favourably with their algorithm.

The remainder of the chapter is organized as follows. Sect. 12.2 presents a new multi-pass scheduling framework incorporating the NP method and the OCBA method. Sect. 12.3 discusses the implementation details. The performance of the proposed framework is assessed through an experimental design in Sect. 12.4. Sect. 12.5 concludes the chapter with a summary and a discussion of directions for future research.

12.2 Multi-pass Scheduling Using Nested Partitions and Optimal Computing Budget Allocation

12.2.1 The Proposed Multi-pass Scheduling Framework

To solve Eq. 12.2, one needs both an effective way to explore the solution space and an efficient way to exploit a selected neighbourhood. To this end, we combine global random search using NP with statistical selection using OCBA. The combined NP+OCBA algorithm is applied to multi-pass scheduling to minimize the number of rule combinations to be evaluated and the number of samples (simulation replications) needed in the evaluation of each rule combination. The flow diagram of the proposed framework is shown in Figure 12.2. Once all the rule combinations are generated, they are partitioned into several disjoint subsets according to the NP method. Then, representative rule combinations are sampled from each set in order to reduce the time required for evaluating all the rule combinations within each set. The sampled rule combinations are evaluated via simulation, through which the best set, the one with the highest likelihood of containing the best rule combination, is selected.

With the OCBA method (innermost loop in Figure 12.2), each sampled rule combination is simulated for a predetermined number of replications to estimate its performance. To ensure that the best rule combination is identified, additional replications must be allocated to the sampled rule combination until the desired probability of correct selection is achieved (or the computational budget is exhausted). Based on the simulation results, the most promising set is selected from among several disjoint sets. If the most promising set is not a singleton, further partitioning is performed and the above procedure is repeated.

12.2.2 The Outer Loop: Nested Partitions

The NP method can be viewed as a metaheuristic, which provides a framework to guide application-specific heuristics by restricting which set of solutions should or can be visited next. It therefore employs a global sampling strategy that is continuously adapted via a partitioning of the feasible regions (Shi and Ólafsson, 2009, Chen and Shi, 2000). The method can also be applied when no closed-form expression exists for the objective function, by estimating the value of the objective function via simulation. In multi-pass scheduling, the feasible region is considered to be the set of all rule combinations, and a rule combination is considered to be a point in the feasible region.

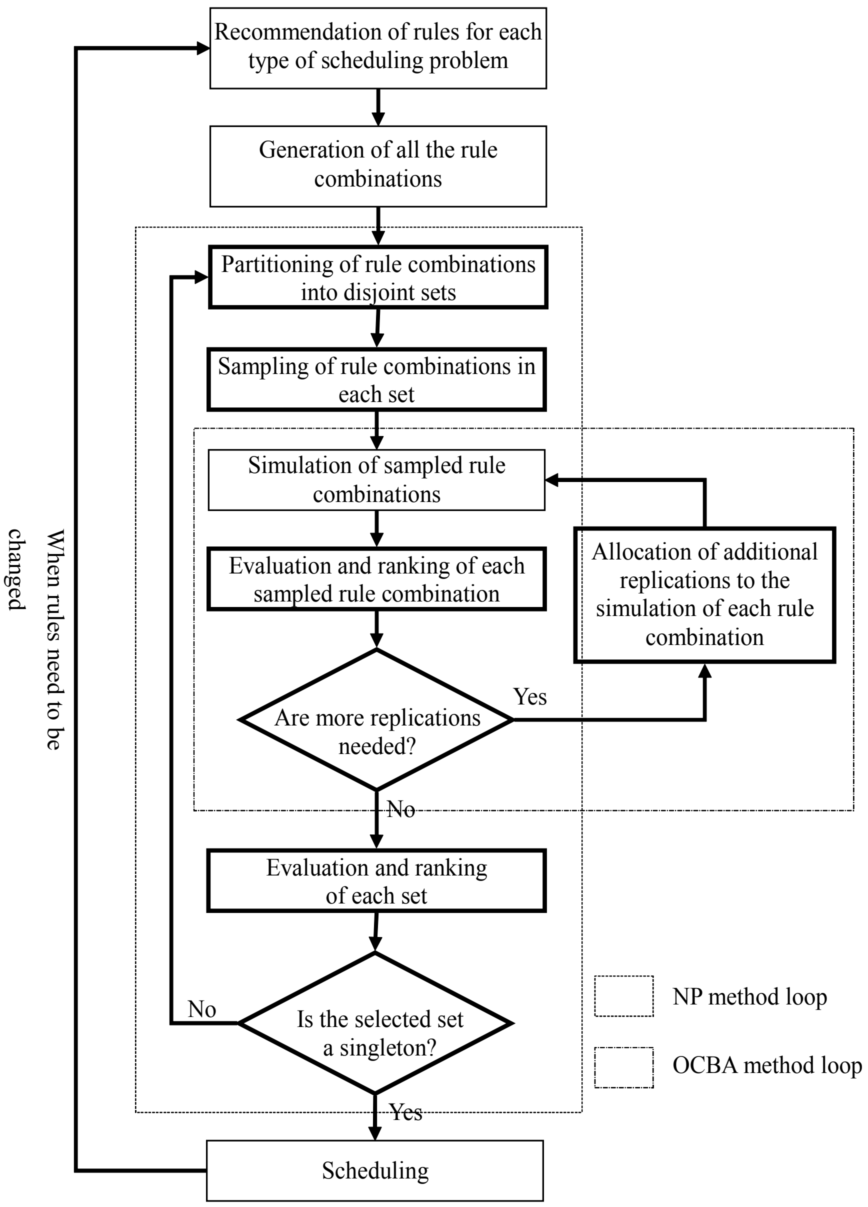

Recommendation of rules for each type of scheduling problem
Generation of all the rule combinations
Partitioning of rule combinations into disjoint sets
Sampling of rule combinations in each set
Simulation of sampled rule combinations
Evaluation and ranking of each sampled rule combination
Allocation of additional replications to the simulation of each rule combination
Are more replications needed?
Yes
No
Evaluation and ranking of each set
Is the selected set a singleton?
No
Yes
Scheduling
When rules need to be changed
NP method loop
OCBA method loop

Fig. 12.2 Detailed flow diagram of proposed multi-pass scheduling framework

As shown in the outer loop of Figure 12.2, the NP method for multi-pass scheduling can be outlined as follows:

- Step 0: the entire feasible set is considered as the most promising set.
- Step 1: the most promising set is partitioned into disjoint subsets, unless it contains a single point.
- Step 2: independent points are sampled from each of these subsets by using some random sampling procedure.
- Step 3: the performance index of each subset is estimated from the collected samples.
- Step 4: the most promising subset is determined by using the estimated index. If two or more subsets are equally promising, the parent subset of the current most promising subset is selected as a feasible set by a backtracking rule.
- Step 5: the selected subset now becomes a feasible set and steps 1 through 5 are repeated for this set.

This procedure generates a sequence of set partitions, with each partition nested within the last one. The final set ultimately contains a single point. The strategies of partitioning, sampling and backtracking deployed for multi-pass scheduling will be described in greater detail after we provide a brief introduction to OCBA.

12.2.3 The Inner Loop: Optimal Computing Budget Allocation

Although the number of rule combinations to be evaluated is reduced at the partitioning stage of the NP method, the computational load may still be high because each rule combination in a partition must be evaluated through a certain number of simulation replications due to the stochastic nature of dynamic scheduling problems. The number of simulation replications needed to guarantee a good estimate is often quite large due to the slow convergence rate of brute-force Monte Carlo simulation. Therefore, to minimize the total number of replications while achieving a desired estimator quality, which will be defined in our case as the correct selection probability, the OCBA method initially assigns an identical number of simulation replications to each sampled rule combination. The rule combinations are then ranked according to these initial simulation results. Additional replications are then allocated to the higher ranked, and therefore potentially critical, rule combinations until the probability of correct selection attains some desired level. (Chen *et al.*, 1997, 1999, 2000ab, 2003). More specifically, OCBA continues in a sequential fashion until the estimated probability that the best rule combination is identified exceeds a threshold or the total simulation budget is exhausted.

As shown in the inner loop of Figure 12.2, the OCBA method can be outlined as follows:

- Step 0: an initial number of replications is allocated to the simulation of each sampled rule combination.
- Step 1: the mean and the variance of the performance of each rule combination are estimated.
- Step 2: it is checked whether additional replications are needed to attain the desired correct selection probability. If no additional replications are needed or if the simulation budget is exhausted, the procedure is terminated.
- Step 3: additional replications for the simulation of the promising rule combinations are performed. Return to step 1.

The deployment of the OCBA method for multi-pass scheduling is discussed next in greater detail.

12.3 Implementation of Nested Partitions and Optimal Computing Budget Allocation

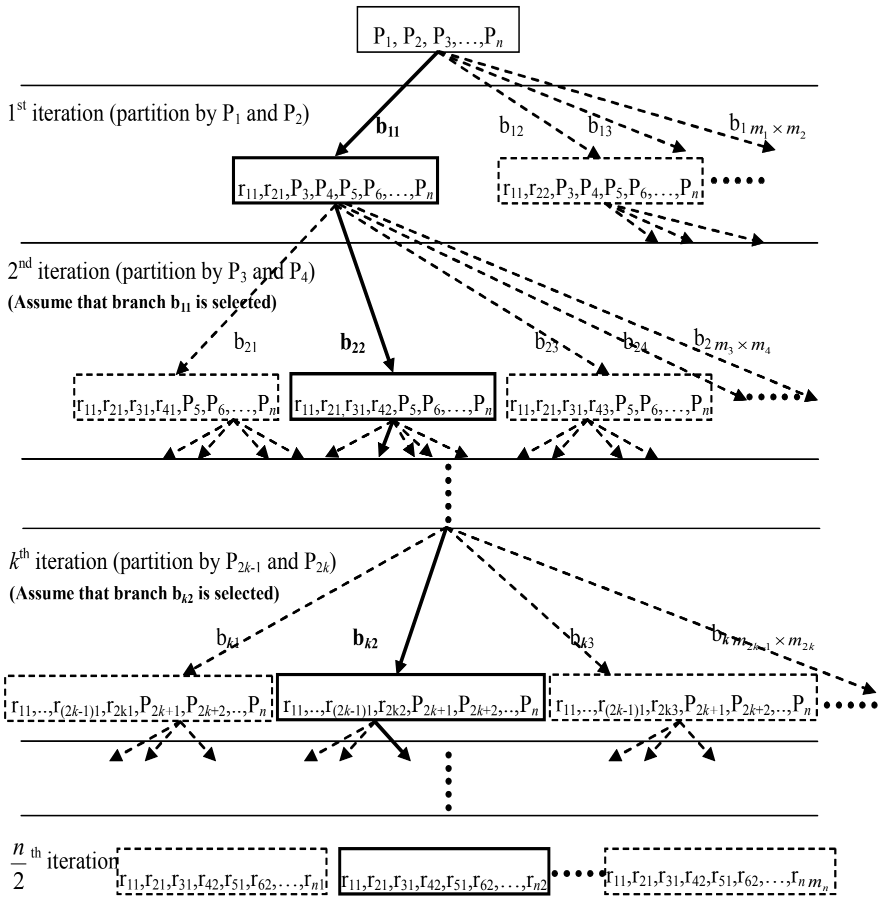
12.3.1 Nested Partitions: Partitioning Strategy

The basic idea of the NP method is to partition the most promising set chosen in the previous iteration into disjoint subsets to be evaluated at the next iteration. The deployed partitioning strategy is therefore crucial for the rapid identification of the best rule combination. Suppose that there exist n different types of scheduling problems and m_i ($i = 1,\ldots, n$) different rules for solving each scheduling problem type. The total number of rule combinations generated and evaluated is then T ($=m_1 \times m_2 \times \cdots \times m_n$). In the first iteration, the feasible region is partitioned into m_1 subsets (i.e., the number of rules for the first problem type). Each subset contains one rule for the first problem type (note that a particular rule does not belong to two different subsets) and all the rule combinations for the other problem types. This implies that all the subsets contain an equal number of rule combinations, that is, $m_2 \times \cdots \times m_n$. Once the most promising subset is chosen in the first iteration, the best rule for the first problem type will have been determined. At the next iteration, the best rule for the second problem type is chosen in the same manner. Thus, for the entire partitioning procedure, the number of iterations, which corresponds to the maximum depth, is n and the number of partitions in iteration i is m_i.

If too many problem types exist, this approach might entail a heavy computational load, which in turn necessitates a more sophisticated partitioning approach. For example, if the number of iterations (i.e., the number of problem types) is large, the number of partitions can be reduced by grouping the problem types through the so-called "knowledge-based clustered partitioning" (Shi *et al.*, 1999). By doing so, the number of iterations (i.e., the depth of the search tree) is decreased but the number of subsets (i.e., partitions) generated at each iteration is increased. For example, when two problem types are grouped, the partitioning strategy consists of the iterations illustrated in Figure 12.3; at the first iteration, problem types P_1 and P_2 are assigned relevant rule combinations. Assuming that partition b_{11} is the most promising one, problem types P_3 and P_4 are assigned rule combinations at the second iteration. At the kth iteration, each subset contains all the rule combinations, in which $2k$ problem types have already been assigned predetermined rule combinations while $n–2k$ problem types have yet to be assigned. The partitioning is repeated until only a single rule combination is left.

P_i: i^{th} problem type
r_{ij}: j^{th} rule in i^{th} problem type
b_{xy}: y^{th} branch in x^{th} iteration
n: the number of problem type

Rule set for problem type 1 (P_1): $\{r_{11}, r_{12}\ldots r_{1m_1}\}$
Rule set for problem type 2 (P_2): $\{r_{21}, r_{22}\ldots r_{2m_2}\}$
: :
Rule set for problem type k (P_k): $\{r_{k1}, r_{k2}\ldots r_{km_k}\}$
: :
Rule set for problem type n (P_n): $\{r_{n1}, r_{n2}\ldots r_{nm_n}\}$



$P_1, P_2, P_3,\ldots,P_n$
1st iteration (partition by P_1 and P_2)
b_{11}
b_{12}
b_{13}
$b_{1m_1\times m_2}$
$r_{11},r_{21},P_3,P_4,P_5,P_6,\ldots,P_n$
$r_{11},r_{22},P_3,P_4,P_5,P_6,\ldots,P_n$
2nd iteration (partition by P_3 and P_4)
(Assume that branch b_{11} is selected)
b_{21}
b_{22}
b_{23}
b_{24}
$b_{2m_3\times m_4}$
$r_{11},r_{21},r_{31},r_{41},P_5,P_6,\ldots,P_n$
$r_{11},r_{21},r_{31},r_{42},P_5,P_6,\ldots,P_n$
$r_{11},r_{21},r_{31},r_{43},P_5,P_6,\ldots,P_n$
k^{th} iteration (partition by P_{2k-1} and P_{2k})
(Assume that branch b_{k2} is selected)
b_{k1}
b_{k2}
b_{k3}
$b_{km_{2k-1}\times m_{2k}}$
$r_{11},..,r_{(2k-1)1},r_{2k1},P_{2k+1},P_{2k+2},..,P_n$
$r_{11},..,r_{(2k-1)1},r_{2k2},P_{2k+1},P_{2k+2},..,P_n$
$r_{11},..,r_{(2k-1)1},r_{2k3},P_{2k+1},P_{2k+2},..,P_n$
$\frac{n}{2}$ th iteration
$r_{11},r_{21},r_{31},r_{42},r_{51},r_{62},\ldots,r_{n1}$
$r_{11},r_{21},r_{31},r_{42},r_{51},r_{62},\ldots,r_{n2}$
$r_{11},r_{21},r_{31},r_{42},r_{51},r_{62},\ldots,r_{nm_n}$

Fig. 12.3 Partitioning strategy of the rule combinations

12.3.2 Nested Partitions: Sampling Strategy

Another issue is the sampling of the rule combinations to evaluate each subset at each iteration since the sampling method and the sample size will greatly affect the sampling accuracy. The relationship between the sampling accuracy and the sample size is convex rather than linear (Chap. 4, Stamatopoulos, 2002); hence, nearly 100% sampling accuracy can be achieved by sampling only half of a subset. Similarly, if the size of each subset is sufficiently large, 30% sampling is sufficient to achieve over 80% sampling accuracy. In general, since the total number of rule combinations (*T*) is typically large, a 30% sampling strategy is adopted in the proposed framework.

With the predetermined sampling size, a two-step sampling strategy is adopted for multi-pass scheduling:

- Step 1: obtain random samples from one subset using uniform sampling with the predetermined sample size.
- Step 2: obtain random samples from the rest of the subsets using a systematic random sampling method.

In multi-pass scheduling, since each partitioned subset has the same size and structure, systematic random sampling can be achieved through uniform random sampling from the rule combinations in the first subset (in step 1) and then by simply picking the same rule combinations in the other subsets (in step 2). Note that the same number of samples is collected from each subset. An example of such a sampling framework is illustrated in Figure 12.4 for three problem types where each problem type has several rules: problem type 1 (A, B, C), problem type 2 (D, E, F, G), and problem type 3 (H, I, J, K, L). Figure 12.4 shows the two-step sampling procedure when the partitioning is performed with the problem type 2 and A is the most promising rule for problem type 1.

12.3.3 Nested Partitions: Backtracking Strategy

The NP method guarantees convergence through backtracking (see Theorems 2.3 and 3.4 in Shi and Ólafsson, 2009). Many backtracking strategies can be considered. In multi-pass scheduling, since "knowledge-based clustered partitioning" is possible, the partitioning strategy and the partitioning history can be traced. If two or more subsets are equally promising, the NP method could backtrack to the immediate superset of the current most promising subset. After the backtracking, the partitioning can be performed based on a different problem type from the problem type that has triggered the backtracking.

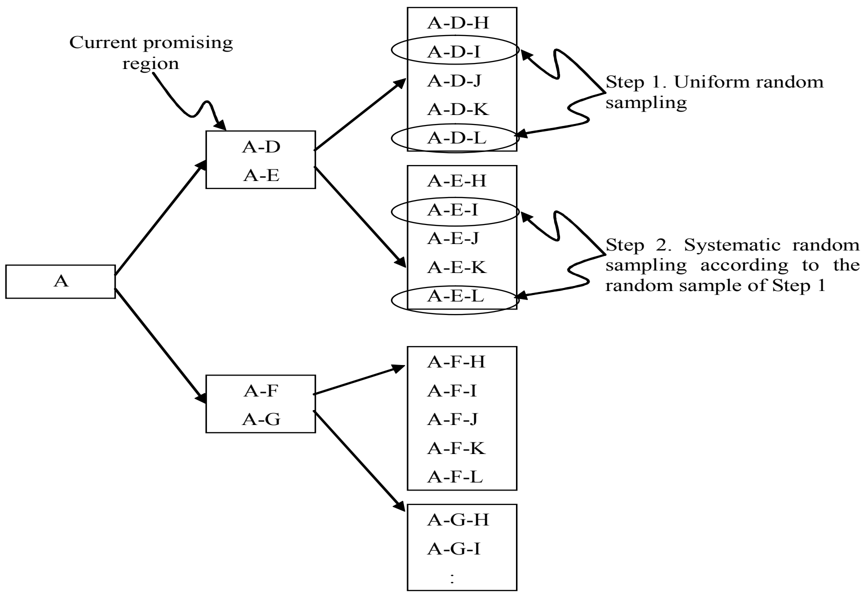


Fig. 12.4 Illustration of sampling strategy

12.3.4 Optimal Computing Budget Allocation: Ranking and Selection Strategy

OCBA aims at determining the optimal number of replications to allot to each rule combination to achieve a desired probability of correct selection ($P\{\mathrm{CS}\}$), where "correct selection" is defined as the event that the selected *best* rule combination *is* actually the best rule combination. Assume that a given partition yields k rule combinations to be evaluated. To this end, we collect N_i IID samples (independent replications or batches) from each rule combination. Let $\boldsymbol{X}_i = \{X_{ij} : j=1,2,\ldots,N_i\}$, $i=1,2,\ldots,k$, be the vector representing the simulation output for rule combination i. We will denote the sample mean performance measure for rule combination i by $\bar{X}_i$ and its variance by s_i^2. Furthermore, let b designate the rule combination with the smallest sample mean performance measure (i.e., $\bar{X}_b \le \min_i \bar{X}_i$) and let s denote the rule combination having the second smallest sample mean performance measure (i.e., $\bar{X}_b \le \bar{X}_s \le \min_{i\neq b} \bar{X}_i$). Let $\delta_{i,j} = \bar{X}_i - \bar{X}_j$. We assume that the simulation output, X_{ij}, has a normal distribution with mean μ_i and variance σ_i^2. After the simulation is performed, a posterior distribution of μ_i is constructed using the prior knowledge on the performance of the rule combination and the observed performance in the simulation output. We can then formally write:

$$P\{CS\} = P\{\overline{X_b} \le \overline{X_i}, i \ne b \mid X_i, i = 1, 2, \ldots, k\} = P\{\hat{\mu}_b \le \hat{\mu}_i, i \ne b\} \tag{12.3}$$

where $\hat{\mu}_i$ denotes the random variable whose probability distribution is the posterior distribution for rule combination i. Since Eq. 12.3 is not directly computable, $P\{\text{CS}\}$ can be bounded through the application of the Bonferroni inequality, which yields an *approximate probability of correct selection* (APCS) (Chen, 1996):

$$P\{CS\} = P\{\bigcap_{i \ne b} [\hat{\mu}_b < \hat{\mu}_i]\} \ge 1 - \sum_{i \ne b} [1 - P\{\hat{\mu}_b < \hat{\mu}_i\}] \equiv APCS\ . \tag{12.4}$$

OCBA is a sequential procedure that quickly identifies inferior alternatives so that additional computing effort is allocated to those alternatives that increase the probability of correct selection. More specifically, at each stage, the optimal number of additional replications can be obtained through the program in Eq. 12.5, where the APCS from Eq. 12.4 is maximized, subject to a constraint on the total additional computing budget:

$$\begin{aligned} &\max \text{APCS} \\ &\text{s.t.} \sum_i N_i = \Delta \end{aligned} \tag{12.5}$$

where Δ is the predefined total additional computing budget at a given stage. A Bayesian approach for solving Eq. 12.5 is presented in Chen *et al.* (1997), where a predictive distribution is constructed to assess the impact of allocating additional simulation replications to competing alternatives on the correct selection probability. The replications are then allocated to those alternatives that achieve the largest anticipated increase in $P\{\text{CS}\}$.

12.3.5 Performance of Nested Partitions and Optimal Computing Budget Allocation

Through the nested partitioning and backtracking strategy, the NP method generates an irreducible recurrent Markov chain whose stationary distribution is used to show the convergence of the method. This is done in Chap. 12 of Shi and Ólafsson (2009), where the most frequently visited solution is used as an estimator of the optimal solution. Further, Shi and Ólafsson (2009) asserted that the convergence proofs are similar regardless of the ranking and selection procedure embedded in the algorithm. Within each partition, therefore, OCBA's fully sequential setup asymptotically improves the APCS by allocating the simulation replications in the following proportions:

$$\frac{N_b}{N_s} \to \frac{\sigma_b}{\sigma_s}\left[\sum_{\substack{i=1\\ i\neq b}}^{k}\left(\frac{\delta_{b,s}^2}{\delta_{b,i}^2}\right)\right]^{1/2},$$
$$\frac{N_i}{N_s} \to \left(\frac{\dfrac{\sigma_i}{\delta_{b,i}}}{\dfrac{\sigma_s}{\delta_{b,s}}}\right),\ i=1,2,\ldots,k;\ i\neq s\neq b. \tag{12.6}$$

The heuristic allocation scheme in Eq. 12.6 appears to be effective in practice, as shown in Chen *et al.* (2000ab) through Chernoff bounds and in Glynn and Juneja (2004) through the large deviations theory. Furthermore, this allocation scheme is quite intuitive: the first equation in Eq. 12.6 implies that a larger number of additional replications should be allocated to the rule combinations that are critical; i.e., those with sample means closer to the best observed performance and those with large estimator noise. Similarly, the second equation in Eq. 12.6 implies that fewer additional replications should be allocated to the non-critical rule combinations. While the above results are concerned with the asymptotic behaviour of optimal computing budget allocation, the finite-time performance of the algorithm is investigated and compared to other techniques in the next section.

12.4 Experimental Design and Analysis

To assess the level of improvement in the effectiveness of multi-pass scheduling, we carried out experiments deploying various approaches, including the NP+OCBA approach, equal allocations, stand-alone OCBA, and COMPASS. As a naïve benchmark, "equal allocation" simulates all rule combinations with an equal computing budget allocation. Stand-alone OCBA allocates simulation replications to all rule combinations without any partitioning of the set of solutions. Finally, COMPASS is based on the algorithm "COMPASS for Fully Constrained DOvS" in Hong and Nelson (2006).

12.4.1 Experimental Assumptions

We will use a full factorial design with m^p experimental runs, where there are m levels of p factors. In particular, we adopt the following assumptions and conditions in our illustrative study:

1. The performance measure is the mean flow time on the shop-floor.

2. There are five different part types on the shop-floor.
3. There are three different machine types on the shop-floor.
4. There are three automated guided vehicle relocation rules after delivery: move to the I/O station, move to the next position, and stay at the current position.
5. There are fix part dispatching rules: SPT (shortest processing fime), EDD (earliest due date), SST (shortest setup time), FIFO (first in first out), COVERT (cost over time), and FOPNR (fewest number of operations remaining).
6. There are five part-releasing rules: SPT, EDD, SST, COVERT, and FOPNR.
7. There are two part-buffering problems after machining: going to the buffer storage and stay at the same machine.
8. The setup time of each machine type is constant.
9. Machine breakdowns are not considered.
10. The routing path and the mean processing time at each machine type are shown in Table 12.1.
11. The mean transportation time between machine types is shown in Table 12.2.
12. The distribution of the processing time and the transportation time is normally truncated to avoid negative realizations.

Note that the objective of this simulation is to find the best rule combination among the 180 possible rule combinations (= 3 × 6 × 5 × 2) as described in 4, 5, 6, and 7 above.

Table 12.1 Routing path and mean processing time

Part type	Routing path	Mean processing time
Part A	Machine type 1 → Machine type 2 → Machine type 3	6.0 – 3.0 – 5.0
Part B	Machine type 3 → Machine type 1 → Machine type 2	7.0 – 3.5 – 4.5
Part C	Machine type 1 → Machine type 3 → Machine type 2	2.0 – 3.0 – 2.0
Part D	Machine type 2 → Machine type 3 → Machine type 1	6.0 – 2.5 – 4.0
Part E	Machine type 2 → Machine type 1 → Machine type 3	5.0 – 3.0 – 3.5

Table 12.2 Mean transportation time

Next location Current location	Machine type 1	Machine type 2	Machine type 3	I/O station
Machine type 1	0.0	1.0	3.0	2.0
Machine type 2	1.0	0.0	2.0	3.0
Machine type 3	3.0	2.0	0.0	4.0
I/O station	2.0	3.0	4.0	0.0

12.4.2 Experimental Design

The controllable factors for the first experiment include the approach of multi-pass scheduling and the number of replications. We considered two levels for multi-pass scheduling (namely, COMPASS and NP+OCBA) while we used 30 levels for the number of replications (namely, 1800, 3600, 5400,…, 54,000). A full factorial design for the controllable factors therefore results in 60 (= 2×30) different experimental base designs.

It is also necessary to define uncontrollable factors over which experiments are performed. Four uncontrollable factors are considered: the simulation window size, the number of machines for each type, the variance of the processing time, and the variance of the transportation time. Each factor consists of three levels. An L9 Taguchi design is used for the uncontrollable factors, which results in nine different conditions (Wu and Michael, 2000).

1. The simulation window size: 150, 300, and 450 events requiring decision-making.
2. The number of machines for each machine type: 1, 2, and 3.
3. The variances of the processing time: 0.01, 0.03, and 0.05 × mean processing time.
4. The variances of the transportation time: 0.01, 0.03, and 0.05 × mean transportation time.

Then, the total number of experiments amounts to 540 (= 60×9). The design matrix for the sampling plan is illustrated in Table 12.3. The rows in the lower left quadrant in the table represent a full factorial experimental design of controllable factors. The columns in the upper right quadrant of the table represent a fractional factorial design of uncontrollable factors. Let $y_{i,j}$ be the average probability of correct selection for a given number of replications. In the experiments, the average ($\bar{y}_i$) of the attained correct selection probability for each design index and its variance (v_{y_i}) are computed as follows:

$$\bar{y}_i = \sum_{j=1}^{n} y_{i,j} \Big/ n, \qquad i = 1,\ldots,m$$

$$v_{y_i} = \sum_{j=1}^{n} (y_{i,j} - \bar{y}_i)^2 \Big/ n-1, \qquad i = 1,\ldots,m \tag{12.7}$$

Then, the robustness r_{yi} of the mean can be defined as follows:

$$r_{y_i} = \log_{10} v_{y_i}, \qquad i = 1,\ldots,m \tag{12.8}$$

Note that a small value indicates higher robustness.

In addition, the signal-to-noise ratio (SNR) can be computed to combine the average performance and its robustness. Since we typically wish to maximize the probability of correct selection, the SNR can be obtained as follows:

$$SNR_l = -10\log_{10}\left(\sum_{j=1}^{n} y_{i,j}^{-2} \Big/ n\right), \qquad i = 1,\ldots,m \tag{12.9}$$

Note that a higher SNR is desirable.

Table 12.3 Design matrix for the sampling plan based on the probability of correct selection (C: COMPASS approach, P: NP+OCBA)

Uncontrollable factor set									
Condition index	1	2	3	4	5	6	7	8	9
Simulation window size	150	300	450	450	150	300	300	450	150
The number of type 1 machines	1	1	1	2	2	2	3	3	3
The number of type 2 machines	1	1	1	2	2	2	3	3	3
The number of type 3 machines	1	1	1	2	2	2	3	3	3
Variance of processing time	0.01	0.03	0.05	0.01	0.03	0.05	0.01	0.03	0.05
Variance of transportation time	0.01	0.03	0.05	0.03	0.05	0.01	0.05	0.01	0.03

Controllable factor set											
Design index	Approach	Replication									
1	C	900	$y_{1,1}$	$y_{1,2}$				...			$y_{1,9}$
2	P	1800	$y_{2,1}$	$y_{2,2}$				...			$y_{2,9}$
3	C	2700	$y_{3,1}$	$y_{3,2}$				...			$y_{3,9}$
4	P	3600	$y_{4,1}$	$y_{4,2}$				...			$y_{4,9}$
:	:	:	:	:				:			:
60	P	54000	$y_{60,1}$	$y_{60,2}$				...			$y_{60,9}$

12.4.3 Experimental Results for the Probability of Correct Selection

The simulation experiments were performed using independent replications. In the experiment of multi-pass scheduling, we also kept track of the sequence of generated partitions with the objective of assessing the effectiveness of NP+OCBA and COMPASS. During the iterations, it was observed that NP+OCBA generated and evaluated only 90 rule combinations from among the full set of 180 rule combinations.

Figure 12.5 summarizes the performance of the four algorithms. We immediately note that the uncontrollable factors do have significant influence on the effectiveness of multi-pass scheduling. For example, in the uncontrollable factor set 1, $P\{\text{CS}\}$ of three algorithms (stand-alone OCBA, COMPASS, and NP+OCBA) is higher than 0.8 even with fewer than 20,000 total replications. Whereas, for the uncontrollable factor set 9, we could not attain the desired $P\{\text{CS}\}$ of 0.9 even with 54,000 total replications. In all but two cases, NP+OCBA dominates the other approaches in that it attains the desired correct selection probability with the smallest number of replications. These results further attest to the value of embedding OCBA into NP as the combined algorithm enhances the effectiveness of stand-alone OCBA.

Finally, as COMPASS and NP+OCBA are identified as the top two performers, we further compare their performances in terms of robustness and signal-to-noise ratio (SNR) through an analysis of variance (ANOVA) study.

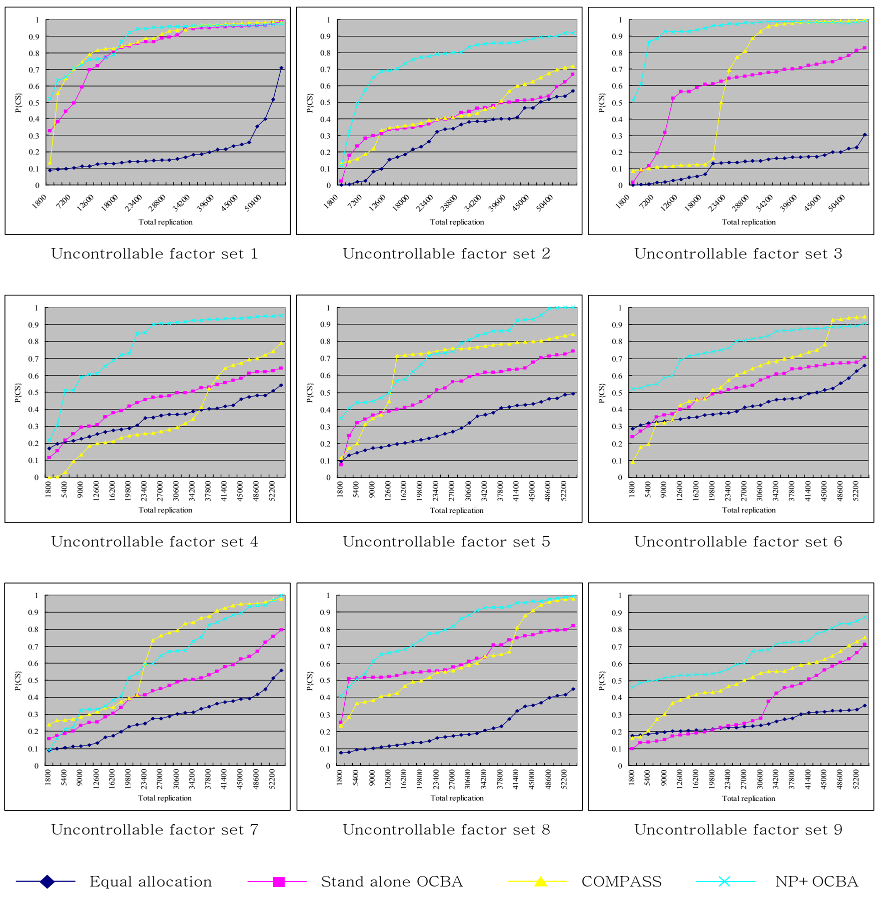


Fig. 12.5 Comparison of $P\{\text{CS}\}$ for each uncontrollable factor set

Table 12.4 shows the average probability of correct selection for a given number of replications for the two approaches with 30 macro simulation runs. Under both approaches, increasing the computing budget (i.e., the number of replications) tends to increase the probability of correct selection. Furthermore, NP+OCBA provides a higher probability of correct selection than COMPASS for a fixed computing budget. When the total computing budget is increased (in our case, beyond 30,000 total replications), the performance of the two approaches becomes indistinguishable.

Table 12.4 Average probability of correct selection of each design (C: COMPASS approach, P: NP+OCBA)

			Uncontrollable factor set								
Condition index			1	2	3	4	5	6	7	8	9
Simulation window size			150	300	450	450	150	300	300	450	150
The num. of mach. type 1			1	1	1	2	2	2	3	3	3
The num. of mach. type 2			1	1	1	2	2	2	3	3	3
The num. of mach. type 3			1	1	1	2	2	2	3	3	3
Var. of process time			0.01	0.03	0.05	0.01	0.03	0.05	0.01	0.02	0.05
Var. of transport time			0.01	0.03	0.05	0.03	0.05	0.01	0.03	0.01	0.03
Controllable factor set											
Design index	App-roach	Repli-cations									
1	C	1800	0.13	0.13	0.08	0.00	0.11	0.09	0.24	0.23	0.16
2	P	1800	0.52	0.11	0.51	0.21	0.34	0.51	0.09	0.41	0.46
3	C	3600	0.55	0.14	0.09	0.00	0.17	0.17	0.26	0.28	0.16
4	P	3600	0.63	0.32	0.61	0.30	0.40	0.52	0.15	0.46	0.48
5	C	5400	0.64	0.15	0.10	0.03	0.20	0.19	0.26	0.36	0.20
6	P	5400	0.65	0.49	0.86	0.51	0.43	0.54	0.21	0.51	0.49
7	C	7200	0.70	0.18	0.10	0.09	0.31	0.31	0.27	0.37	0.27
8	P	7200	0.70	0.57	0.88	0.51	0.44	0.54	0.22	0.53	0.50
9	C	9000	0.74	0.22	0.11	0.13	0.34	0.32	0.28	0.38	0.30
10	P	9000	0.72	0.65	0.92	0.59	0.44	0.58	0.32	0.61	0.51
11	C	10800	0.79	0.33	0.11	0.18	0.36	0.34	0.29	0.40	0.36
12	P	10800	0.75	0.68	0.92	0.60	0.46	0.60	0.33	0.65	0.52
13	C	12600	0.81	0.34	0.12	0.20	0.44	0.42	0.31	0.41	0.38
14	P	12600	0.76	0.69	0.92	0.61	0.49	0.69	0.33	0.66	0.53
15	C	14400	0.82	0.35	0.12	0.20	0.71	0.44	0.33	0.42	0.40
16	P	14400	0.76	0.70	0.92	0.65	0.56	0.71	0.34	0.67	0.53
17	C	16200	0.82	0.36	0.12	0.21	0.72	0.45	0.34	0.46	0.41
18	P	16200	0.78	0.73	0.93	0.68	0.57	0.72	0.38	0.68	0.53
19	C	18000	0.83	0.36	0.12	0.23	0.72	0.46	0.37	0.49	0.42

Table 12.4 (continued)

Controllable factor set											
Design index	App-roach	Repli-cations									
20	P	18000	0.86	0.75	0.94	0.72	0.62	0.73	0.40	0.70	0.53
21	C	19800	0.84	0.37	0.16	0.24	0.72	0.51	0.39	0.49	0.43
22	P	19800	0.92	0.77	0.96	0.73	0.66	0.74	0.51	0.73	0.54
23	C	21600	0.86	0.39	0.49	0.25	0.73	0.53	0.41	0.51	0.44
24	P	21600	0.94	0.77	0.96	0.84	0.71	0.75	0.54	0.77	0.54
25	C	23400	0.88	0.39	0.69	0.25	0.74	0.57	0.60	0.54	0.46
26	P	23400	0.94	0.79	0.97	0.85	0.72	0.76	0.59	0.77	0.56
27	C	25200	0.88	0.40	0.77	0.26	0.75	0.60	0.73	0.55	0.47
28	P	25200	0.95	0.79	0.97	0.90	0.73	0.80	0.59	0.79	0.59
29	C	27000	0.91	0.41	0.81	0.27	0.75	0.62	0.76	0.55	0.50
30	P	27000	0.95	0.80	0.98	0.90	0.73	0.80	0.64	0.81	0.60
31	C	28800	0.93	0.41	0.88	0.28	0.75	0.64	0.78	0.57	0.51
32	P	28800	0.95	0.80	0.97	0.90	0.79	0.81	0.66	0.86	0.67
33	C	30600	0.93	0.42	0.92	0.29	0.76	0.66	0.79	0.59	0.54
34	P	30600	0.96	0.83	0.98	0.91	0.80	0.82	0.67	0.88	0.67
35	C	32400	0.94	0.43	0.96	0.31	0.76	0.67	0.83	0.60	0.55
36	P	32400	0.96	0.84	0.98	0.91	0.83	0.83	0.67	0.90	0.68
37	C	34200	0.96	0.46	0.96	0.34	0.77	0.68	0.84	0.64	0.55
38	P	34200	0.96	0.85	0.98	0.92	0.84	0.86	0.73	0.92	0.71
39	C	36000	0.96	0.47	0.97	0.41	0.77	0.69	0.86	0.64	0.55
40	P	36000	0.96	0.85	0.98	0.92	0.86	0.86	0.75	0.92	0.72
41	C	37800	0.96	0.50	0.97	0.52	0.78	0.70	0.87	0.65	0.57
42	P	37800	0.96	0.85	0.98	0.92	0.86	0.86	0.82	0.92	0.72
43	C	39600	0.97	0.57	0.98	0.58	0.78	0.72	0.90	0.66	0.59
44	P	39600	0.96	0.86	0.98	0.93	0.86	0.87	0.84	0.93	0.72
45	C	41400	0.97	0.59	0.98	0.64	0.79	0.73	0.92	0.80	0.60
46	P	41400	0.96	0.86	0.98	0.93	0.92	0.87	0.86	0.95	0.73
47	C	43200	0.97	0.60	0.98	0.66	0.79	0.75	0.94	0.87	0.60
48	P	43200	0.96	0.87	0.98	0.93	0.92	0.87	0.88	0.95	0.77
49	C	45000	0.97	0.62	0.98	0.67	0.80	0.78	0.94	0.91	0.62
50	P	45000	0.96	0.88	0.98	0.93	0.93	0.88	0.89	0.96	0.78
51	C	46800	0.98	0.65	0.98	0.69	0.80	0.92	0.95	0.94	0.64
52	P	46800	0.96	0.89	0.98	0.94	0.95	0.88	0.93	0.96	0.80
53	C	48600	0.98	0.67	0.99	0.69	0.81	0.93	0.95	0.96	0.67
54	P	48600	0.96	0.89	0.98	0.94	0.99	0.88	0.93	0.97	0.83
55	C	50400	0.98	0.69	0.99	0.72	0.82	0.94	0.96	0.97	0.70
56	P	50400	0.97	0.90	0.97	0.94	0.99	0.89	0.94	0.98	0.83
57	C	52200	0.98	0.70	0.99	0.74	0.83	0.94	0.97	0.97	0.72
58	P	52200	0.97	0.91	0.98	0.95	0.99	0.89	0.97	0.98	0.84
59	C	54000	0.98	0.71	0.99	0.79	0.84	0.94	0.98	0.97	0.75
60	P	54000	0.97	0.91	0.99	0.95	0.99	0.90	0.99	0.99	0.87

Table 12.5 summarizes the experimental results. Note that smaller values for r_{y_i} and larger values for SNR indicate a more robust approach. In particular, pairwise comparisons (i.e., between design indices 1 and 2, between design indices 3 and 4, etc.) reveal a higher robustness and SNR for NP+OCBA. As we compute

robustness with respect to the uncontrollable factor set, we can conclude that NP+OCBA is more robust than COMPASS in the tested shop-floor environment.

Table 12.5 Experiment results for average probability of correct selection

Controllable factor set			Average $\left(\overline{y}_i\right)$	Robustness $\left(r_{y_i}\right)$	SNR
Design index	Approach	Replications			
1	C	1800	0.13233	–2.2565	–50.46
2	P	1800	0.35542	–1.5244	–14.372
3	C	3600	0.207	–1.6182	–38.437
4	P	3600	0.43544	–1.6275	–9.6861
5	C	5400	0.24144	–1.4962	–21.75
6	P	5400	0.52487	–1.5239	–7.4702
7	C	7200	0.29289	–1.4875	–14.972
8	P	7200	0.54784	–1.4917	–7.0806
9	C	9000	0.31633	–1.4675	–13.612
10	P	9000	0.59844	–1.5317	–5.5331
11	C	10800	0.35567	–1.4514	–12.404
12	P	10800	0.61698	–1.5324	–5.2578
13	C	12600	0.38567	–1.4258	–11.765
14	P	12600	0.63348	–1.5372	–5.0466
15	C	14400	0.42656	–1.3061	–11.469
16	P	14400	0.65418	–1.5785	–4.6523
17	C	16200	0.43622	–1.3076	–11.278
18	P	16200	0.6718	–1.6012	–4.2677
19	C	18000	0.44978	–1.3073	–11.01
20	P	18000	0.69941	–1.5754	–3.899
21	C	19800	0.46633	–1.3308	–9.7543
22	P	19800	0.7323	–1.65	–3.2184
23	C	21600	0.51589	–1.4681	–7.2608
24	P	21600	0.76318	–1.6501	–2.8535
25	C	23400	0.574	–1.4474	–6.5195
26	P	23400	0.77673	–1.718	–2.5949
27	C	25200	0.60522	–1.3957	–6.2096
28	P	25200	0.79371	–1.7239	–2.3904
29	C	27000	0.62256	–1.3665	–5.9644
30	P	27000	0.80572	–1.7722	–2.2045
31	C	28800	0.644	–1.3288	–5.6641
32	P	28800	0.82856	–1.9131	–1.852
33	C	30600	0.65911	–1.318	–5.4039
34	P	30600	0.83871	–1.9074	–1.7507
35	C	32400	0.67656	–1.3084	–5.0456
36	P	32400	0.84918	–1.9114	–1.641
37	C	34200	0.69167	–1.3284	–4.6583
38	P	34200	0.86707	–2.0424	–1.3912
39	C	36000	0.708	–1.3711	–4.1095
40	P	36000	0.87299	–2.0953	–1.3127
41	C	37800	0.73056	–1.478	–3.4536
42	P	37800	0.88292	–2.1999	–1.1841
43	C	39600	0.754	–1.5574	–2.9946
44	P	39600	0.88653	–2.2069	–1.1474

Table 12.5 (continued)

Controllable factor set			Average $(\bar{y}_i)$	Robustness (r_{y_i})	SNR
Design index	Approach	Replications			
45	C	41400	0.78533	−1.6288	−2.55
46	P	41400	0.89976	−2.2279	−1.0143
47	C	43200	0.80078	−1.632	−2.3734
48	P	43200	0.90871	−2.3852	−0.89481
49	C	45000	0.81467	−1.6566	−2.1896
50	P	45000	0.914	−2.4386	−0.83629
51	C	46800	0.84244	−1.6736	−1.8755
52	P	46800	0.92531	−2.5289	−0.71803
53	C	48600	0.85311	−1.7173	−1.7159
54	P	48600	0.93536	−2.5468	−0.62034
55	C	50400	0.86578	−1.7794	−1.5311
56	P	50400	0.93841	−2.5431	−0.59221
57	C	52200	0.87667	−1.8231	−1.388
58	P	52200	0.94659	−2.5881	−0.51233
59	C	54000	0.88822	−1.9045	−1.2275
60	P	54000	0.95628	−2.6569	−0.41743

Finally, we confirm our observations through an ANOVA study. In particular, the following hypothesis was tested at significance level 0.05:

H_0: $P\{CS\}_{NP+OCBA} = P\{CS\}_{COMPASS}$

H_1: $P\{CS\}_{NP+OCBA} > P\{CS\}_{COMPASS}$

The ANOVA results, summarized in Table 12.6, show that the results obtained using NP+OCBA are significantly better than those obtained using COMPASS for smaller computational budgets, namely until design 35&36 (with a total budget of up to 32,400 total replications, and with an F statistic > 4.49, a p-value < 0.05). As indicated before, when the computational budget becomes sufficiently large, the difference in performance between NP+OCBA and COMPASS vanishes.

Table 12.6 ANOVA test of the correct selection probability

Design	Replications	F-value	P-value
1&2	1800	12.64162	0.00264
3&4	3600	9.85401	0.00634
5&6	5400	11.69353	0.00351
7&8	7200	9.03112	0.00839
9&10	9000	11.28439	0.00399
11&12	10800	9.49644	0.00715
13&14	12600	8.30604	0.01084
15&16	14400	6.15048	0.02464
17&18	16200	6.72254	0.01963
19&20	18000	7.39291	0.01517
21&22	19800	9.21613	0.00787
23&24	21600	9.75645	0.00655
25&26	23400	6.74549	0.01945
27&28	25200	5.41069	0.03348
29&30	27000	5.04111	0.03924

Table 12.6 (continued)

Design	Replications	F-value	P-value
31&32	28800	5.18558	0.03686
33&34	30600	4.80187	0.04357
35&36	32400	4.36663	0.05298
37&38	34200	4.94289	0.04095
39&40	36000	4.84386	0.04277
41&42	37800	5.27935	0.03541
43&44	39600	4.66122	0.04638
45&46	41400	4.00427	0.06265
47&48	43200	3.81888	0.06839
49&50	45000	3.45644	0.08150
51&52	46800	2.55781	0.12931
53&54	48600	2.76532	0.11579
55&56	50400	2.43699	0.13806
57&58	52200	2.49857	0.13351
59&60	54000	2.84259	0.11119

12.5 Conclusions

For OvS, it is necessary to design both an effective algorithm to explore the solution space and an efficient algorithm to exploit a selected neighbourhood. To this end, we combine global random search using NP with statistical selection using OCBA. As a non-trivial illustration, the combined NP+OCBA algorithm is applied to the multi-pass scheduling problem. Multi-pass scheduling evaluates through simulation multiple courses of action before actual execution and then, based on the simulation results, selects the best course of action. Although multi-pass scheduling performs much better than single-pass scheduling, it suffers from the disadvantage that it entails a significantly heavy computational load in evaluating all possible candidate rules. In this chapter, a new multi-pass scheduling framework was presented that minimizes the number of rules to be evaluated by using an NP method, and minimizes the total number of simulation replications by using an OCBA method. The NP method utilizes the partitioning of the feasible rules and the sampling of each partition. The OCBA method allocates more replications to potentially critical rules. The combined NP+OCBA algorithm can be embedded in a manufacturing execution system (MES). Upon the occurrence of an event requiring decision-making, the MES can call the NP+OCBA module to evaluate the relevant rule combinations and rapidly determine the best option to pursue.

Finally, the efficiency and effectiveness of the proposed approach were demonstrated by comparing its performance with that of other popular approaches. The results show that NP+OCBA provides more accurate results with a smaller computational effort. Furthermore, given its robustness, NP+OCBA appears to be better suited for real-time shop-floor scheduling and control.

OvS is a significant step not only in increasing the efficiency of multi-pass scheduling, but also in enhancing the role of simulation as a decision-support tool in more general settings. Current research focuses on the development and deployment other OvS techniques to take full advantage of the modelling flexibility of simulation and algorithmic efficiency of optimization.

Acknowledgements This research has been sponsored by the Korean Ministry of Science and Technology and the French Embassy in Seoul, Korea.

References

Alrefaei MH, Andradóttir S (1999) A simulated annealing algorithm with constant temperature for discrete stochastic optimization. Mgment. Sci., 45:748–764

Alrefaei MH, Andradóttir S (2001) A modification of the stochastic ruler method for discrete stochastic optimization. Eur. J. Oper. Res., 133:160–182

Andradóttir S (1998) Simulation optimization. In: J. Banks (ed.) Handbook of Simulation, John Wiley, New York

Chen CH (1996) A lower bound for the correct subset-selection probability and its application to discrete-event system simulations. IEEE Trans. Auto. Control, 41(8):1227–1231

Chen CH, Shi L (2000) A new algorithm for stochastic discrete resource allocation optimization. Disc. Event Dynam. Syst.: Theo. Appl., 10:271–294

Chen CH, Yücesan E, Chen HC et al. (1997) New development of optimal computing budget allocation for discrete event simulation. In: Proceedings of the 1997 Winter Simulation Conference, pp. 334–341

Chen CH, Wu D, Dai L (1999) Ordinal comparison of heuristic algorithms using stochastic optimization. IEEE Trans. Robo. Auto., 15(1):44–56

Chen CH, Lin J, Yücesan E et al. (2000a) Simulation budget allocation for further enhancing the efficiency of ordinal optimization. J. Disc. Event. Dynam. Syst., 10:251–270

Chen HC, Chen CH, Yücesan E (2000b) Computing efforts allocation for ordinal optimization and discrete event simulation. IEEE Trans. Auto. Control, 45:960-964

Chen CH, Yüesan E, Lin J et al. (2003) Optimal computing budget allocation for Monte-Carlo simulation with application to product design. Simu. Mode. Prac. Theo., 11:57–74

Chiu C, Yih Y (1995) A learning-based methodology for dynamic scheduling in distributed manufacturing systems. Int. J. Prod. Res., 33(11):3217–3232

Cho H, Wysk RA (1993) A robust adaptive scheduler for an intelligent workstation controller. Int. J. Prod. Res., 31:771-789

Chunda B, Mize JH (1994) Scheduling and control of flexible manufacturing systems: a critical review. Int. J. Comput. Integr. Manuf., 7:340–355

Davis WJ, Jones AT (1988) A real-time production scheduler for a stochastic manufacturing environment. Int. J. Comput. Integr. Manuf., 4(4):531–544

Fu MC (2002) Optimization for simulation: theory and practice. INFORMS J. Comput., 14:192–215

Glynn P, Juneja S (2004) A large deviations perspective on ordinal optimization. In: Proceedings of the 2004 Winter Simulation Conference, pp. 577–585

Gong WB, Ho YC, Zhai W (1999) Stochastic comparison algorithm for discrete optimization with estimation. SIAM J. Opt., 10:384–404

Hong LJ, Nelson BL (2006) Discrete optimization via simulation using COMPASS. Oper. Res., 54:115–129

Jeong KC, Kim YD (1998) A real-time scheduling mechanism for a flexible manufacturing system: using simulation and dispatching rules. Int. J. Prod. Res., 36:2609–2626

Jones AT, Rabelo L, Yih Y (1995) A Hybrid approach for real-time sequencing and scheduling. Int. J. Comput. Integr. Manuf., 8(2):145–154

Kim CO, Min HS, Yih Y (1998) Integration of inductive learning and neural networks for multi-objective FMS scheduling. Int. J. Prod. Res., 36(9):2497–2509

Kutanoğlu E, Sabuncuoğlu I (2002) Experimental investigation of iterative simulation-based scheduling in a dynamic and stochastic job shop. J. Manuf. Syst., 20(4):264–279

Ólafsson S (2004) Two-stage nested partitions method for stochastic optimization. Metho. Comput. Appl. Prob., 6:5–27

Pichitlamken J, Nelson BL, Hong LJ (2006) A sequential procedure for neighbourhood selection of the best in optimization via simulation. Eur. J. Oper. Res., 173:283–296

Rubinstein RY (1999) The cross entropy method for combinatorial and continuous optimization. Metho. Comput. Appl. Prob., 1:127–190

Shi L, Olafsson S (2000) Nested partitions method for stochastic optimization. Meth. Comput. Appl. Prob., 2(3):271–291

Shi L, Olafsson S (2009) Nested partitions: method, theory and applications. Springer

Shi L, Olafsson S, Sun N (1999) New parallel randomized algorithms for the travelling salesman problem. Comput. Oper. Res., 26:371–394

Stamatopoulos C (2002) Sample-based fishery surveys: A technical handbook, FAO, Rome

Sudiarso A, Labib AW (2002) A fuzzy logic approach to an integrated maintenance and production scheduling algorithm. Int. J. Prod. Res., 40(13):3121–3138

Swisher JR, Jacobson SH, Yücesan E (2003) Discrete-event simulation optimization using ranking, selection, and multiple comparison procedures: a survey. ACM Trans. Mod. Comput. Simul., 13:134–154

Tormos P, Lova A (2003) An efficient multi-pass heuristic for project scheduling with constrained resources. Int. J. Prod. Res., 41(5):1071–1086

Wu CFJ, Michael H (2000) Experiments: planning, analysis, and parameter design optimization. John Wiley and Sons

Wu DS, Wysk RA (1988) Multi-pass expert control system a control/scheduling structure for flexible manufacturing cells. J. Manuf. Syst., 7(2):107–120

Yang H, Wu Z (2003) The application of adaptive genetic algorithms in flexible manufacturing systems dynamic rescheduling. Int. J. Comput. Integr. Manuf., 16(6):382–397

Chapter 13
Intelligent Techniques for Safety Stock Optimization in Networked Manufacturing Systems

B. Desmet[1], E.-H. Aghezzaf[2] and H. Vanmaele[1]

Abstract This chapter discusses some intelligent techniques for the solution of the safety stock optimization problem in networked manufacturing systems. These solutions techniques are based on normal approximation models for the involved critical safety stock parameters. The first and second sections introduce the issue of multi-echelon inventory control and review the related literature. The next three sections discuss safety stock optimization in distribution systems, assembly systems and then in generic networked manufacturing systems. The proposed approximation models are tested on small example systems and are benchmarked with results obtained from discrete-event simulation. As the various simulations show, these proposed approximations prove to be rather conservative and provide good upper bounds on the required system safety stocks.

Keywords Safety stock, optimization, distribution systems, assembly systems, generic networked manufacturing systems

13.1 Introduction

Carrying safety stocks is a common hedging strategy to deal with uncertainties stemming from critical parameters such as customer demands and lead times. This chapter discusses the issue of optimizing total safety stock in networked manufacturing systems. In particular, the question of how much safety stock should be carried at each level of the network is investigated. A common strategy one usually comes across in practice consists in carrying safety stocks for finished products in

[1] B. Desmet and H. Vanmaele
MÖBIUS Business Redesign, Kortrijksesteenweg 152, B-9830 Sint-Martens-Latem, Belgium

[2] E.-H. Aghezzaf (✉)
Department of Industrial Management, Faculty of Engineering, Ghent University, Technologiepark Zwijnaarde 903, B-9052 Zwijnaarde, Belgium
e-mail: ElHoussaine.Aghezzaf@UGent.be

the face of demand and lead-time uncertainty. The safety stock levels are then determined in terms of demand and lead-time probability distributions. However, the idea that inventories can be held in forms other than finished products has proven to be very effective in terms of safety stocks and service levels optimization. There are good reasons to hold inventory of semi-finished products at earlier manufacturing stages to hedge not only the external but also internal uncertainties. Optimizing safety stocks in a networked manufacturing system is a challenging task, however. The resulting optimization problems become quickly very complex. Determining the suitable safety stock levels and placements, even in simplest network structures, is fairly difficult and requires some ingenuity.

Traditionally, two approaches are used to optimize safety stock levels in complex systems. In the first approach the entire system is decoupled into "independent" single facilities and determines the safety stock level for each facility independently. One can agree without any difficulty that this approach overestimates the required safety stock levels throughout the system. In the second approach, the safety stock level at a given facility is determined by taking into account existing downstream safety stock levels as well. Of course this approach provides better estimate of the required safety stock levels. However, the resulting safety stock optimization models are very difficult, especially, when the network structure is complex. Therefore, it is important to start the analysis with the simplest networked manufacturing system (i.e. a serial system) and then extend the analysis to a more general networked assembly and distribution systems.

In a serial manufacturing system each facility has at most one predecessor and at most one successor. These systems, also called linear systems, can serve as a step-up to the other more complex systems. Networked distribution systems are typically divergent and each stock point has at most one immediate predecessor. However, assembly networked manufacturing systems are convergent networks in which each facility has at most one successor. Assembly and distribution networks are special cases of spanning tree networked manufacturing systems. A spanning tree structure captures a large variety of real-world networked manufacturing systems. In this chapter we will discuss and propose intelligent solution techniques to optimize safety stocks in each structure starting from serial systems and then extending the analysis to more general networked manufacturing systems.

The remainder of the chapter is organized as follows: Sects. 13.2 and 13.3 introduce and discuss multi-echelon inventory control systems and the related literature. Sect. 13.4 focuses on safety stock modelling in multi-echelon networks. Sect. 13.5 considers safety stock optimization in distribution systems. Sect. 13.6 extends this result to assembly and generic systems. Finally, Sect. 13.7 concludes the chapter.

13.2 Multi-echelon Inventory Control in Networked Manufacturing Systems

The simplest multi-echelon inventory system is a *serial system*. In a serial system each installation has at most one predecessor and at most one successor. An example is shown in Figure13.1. In many treatments the linear system serves as a step-up to more complex systems.

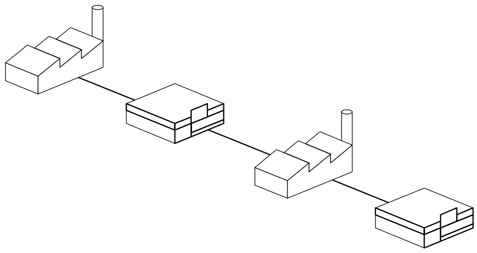


Fig. 13.1 A linear inventory system

Distribution systems are, in general, divergent. In a pure *divergent system*, each stock point has at most one immediate predecessor. Figure13.2 shows such a system with two levels: a central distribution center (DC), and three regional distribution centers (RDC).

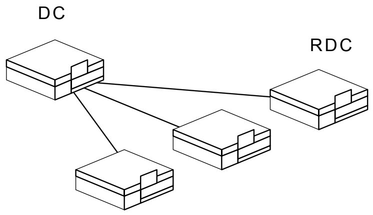


Fig. 13.2 A divergent/distribution inventory system

Inventory systems may also have a *convergent* or *assembly* structure, in which each installation has at most one successor. This is illustrated in Figure 13.3.

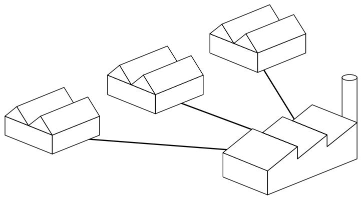


Fig. 13.3 A convergent/assembly inventory system

Assembly and distribution networks are special cases of a spanning tree network. A spanning tree is a connected graph that contains *N* nodes and *N*– 1 arc. It allows capturing numerous real-life supply chains.

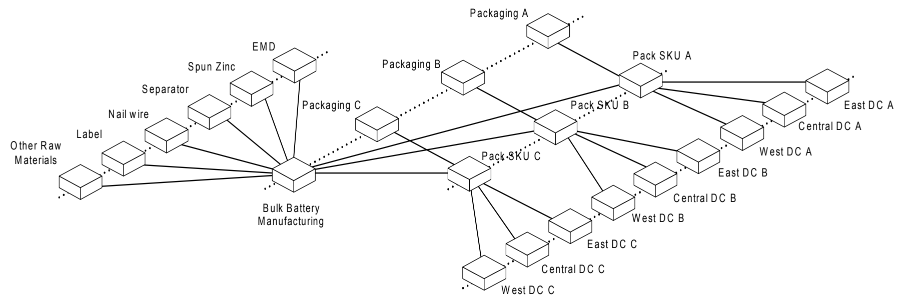


Fig. 13.4 A spanning tree inventory system

Figure 13.4 shows an example of a spanning tree system described and analyzed by Graves and Willems (2003). It considers part of a battery supply chain. The first step is the manufacturing of bulk batteries. This is an assembly type of operation with a convergent structure. The second step is the packaging of the bulk batteries in three different packaging types. This is again an assembly type of operation with a convergent structure. The last step is the distribution of the packed stock-keeping units (SKUs) to the east, central and west DC. This is a distribution operation with a divergent structure. The complete manufacturing and distribution operation has a spanning tree network structure. It has twenty-two nodes and twenty-one arcs.

Some real-life networked manufacturing systems do not qualify as spanning trees. We will refer to them as "generic" systems. The following figure shows a system with five nodes and six arcs.

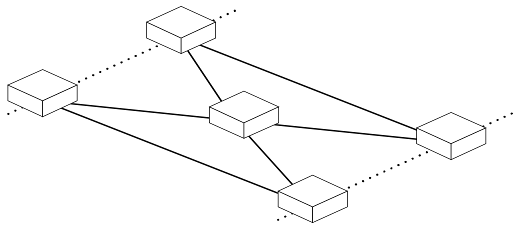


Fig. 13.5 A generic inventory system

Multi-echelon inventory control extends the basic questions of single-echelon inventory control to multi-echelon systems. Figure 13.6 summarizes some of these basic questions for the battery supply chain shown in Figure 13.4. It shows that the average inventory in a multi-echelon system is the sum of different types of inventory on different levels of the system.

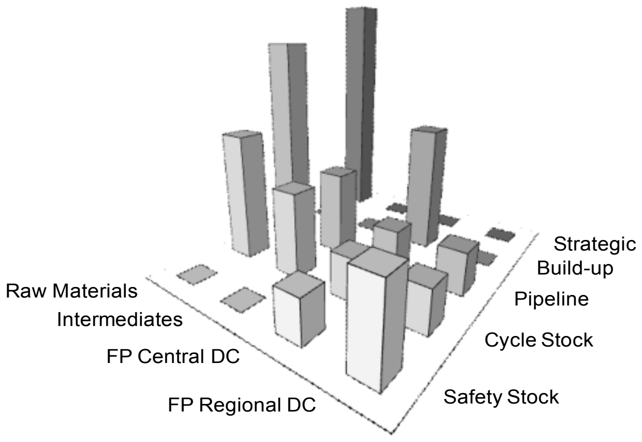


Fig. 13.6 Multi-echelon view of inventory control

Each of the inventory types has its own reason for existence and as such requires a proper control.

- Cycle stock results from the requirement to produce or ship in batches. Reducing the cycle stock typically requires reducing ordering or setup costs or synchronizing operations in the network.
- The safety stock is the buffer against uncertainty in both supply and demand. The level of the safety stock is determined by the level of uncertainty and the desired level of service. Reducing safety stocks requires a reduction of the underlying uncertainties or service levels. In a multi-echelon system a key question is how to pool the risk into a limited number of stock points.
- Pipeline stock is work in progress or transit and is due to non-negative lead times in manufacturing and distribution. They are to be reduced via a reduction of the lead times.
- Buildup stock is typically due to seasonal demands and level production strategies. Extra capacity and an improved planning can typically help minimizing the required buildup.
- Strategic stock tries to anticipate supply problems of critical components. They are a buffer against uncertain but high-impact events and typically depend on a management decision.

The “raw materials” are the components required for the bulk battery manufacturing and the packaging material. The intermediates are the batteries in bulk. The packaged batteries make up the finished product inventory that is stored both centrally in the plant and in the regional DCs. For each of those process steps there is a control question for strategic, buildup, pipeline, safety and cycle stock. Long supply lead times and quantity rebates may result in high pipeline and cycle stock for raw materials. Management may decide building a strategic inventory for some critical components. To anticipate a high season the company may decide to build-up a stock of finished product in the central DC. Finally a multi-echelon safety stock optimization may lead to a safety stock buffer spread over the last two echelons.

Techniques from the domain of "independent single-echelon" can be suboptimal when applied in a multi-echelon context. This is illustrated in the following two outlines and the "simulation game" introduced in the next paragraph. The first outline takes the well-known "economic order quantity" (EOQ) and applies it to a two-echelon linear supply chain. We show it is suboptimal and reveal a surprising way to find a better solution. The second outline illustrates how an ordering policy that makes sense from a single-echelon perspective has adverse effects when applied in a multi-echelon context.

A Single-echelon Pitfall in Lot Size Optimization

A common "single-echelon" lot size calculation is done with the EOQ. In general, increasing lot size decreases setup costs but increases inventory-carrying costs. The EOQ minimizes the combined setup and inventory cost. The EOQ considers a single inventory point. When considering the EOQ in a multi-echelon context, the question is whether optimizing all points in isolation will lead to an optimum for the system. A simple example shows that it does not.

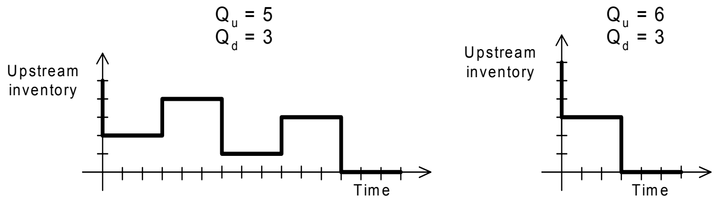


Fig. 13.7 Lot size synchronization in a linear inventory system

Consider a two-echelon linear system. Suppose inventory carrying and setup costs are such that the EOQ downstream is 3 and the upstream is 5. This could be the case if inventory-carrying costs are comparable over the two echelons (not much added value) but setup costs are considerably higher upstream. Suppose the downstream consumption is stationary at one piece per unit of time. It is easy to see that the upstream inventory will follow a cycle of 15 periods, the least common multiple of 3 and 5. The upstream inventory will take all values between 0 and 4 and will be 2 on average. Increasing the upstream lot size to 6 reduces the setup cost as we increase the lot size. It also allows for a reduction in average upstream inventory. The new upstream inventory cycle will be 6 periods, the least common multiple of 3 and 6. The upstream inventory will take two values, 3 and 0, leading to an average of 1.5. The graph in Figure 13.7 summarizes the two scenarios. Multi-echelon lot size models incorporate this kind of effect and provide means to minimize the system's inventory carrying and setup costs. They may lead to dynamics that are counter-intuitive at first, e.g. the above example showing that one can decrease the inventory by increasing the lot size.

A Single-echelon Pitfall in Reordering Policies

Slack *et al.* (2001) present a good illustration of how an ordering policy, that makes sense from a single-echelon perspective, can have a devastating effect in a multi-echelon environment. Consider an order-up-to-policy where the order-up-to-level is a one-month forecast. Suppose that we use a simple forecast technique assuming next month demand will equal demand of this month (moving average considering one period of history). Table 13.1 illustrates how in a linear supply chain, a minor disruption in consumer demand, gets amplified as it passed on to the upstream tiers. It's an illustration of the well-known bullwhip effect.

Table 13.1 Demand amplification in a linear inventory system

	Third-tier supplier		Second-tier supplier		First-tier supplier		Original equipment manufacturer		
Period	Prod.	Stock	Prod.	Stock	Prod.	Stock	Prod.	Stock	Demand
1	100	100	100	100	100	100	100	100	100
		100		100		100		100	
2	20	100	60	100	80	100	90	100	95
		60		80		90		95	
3	180	60	120	80	100	90	95	95	95
		120		100		95		95	
4	60	120	90	100	95	95	95	95	95
		90		95		95		95	
5	100	90	95	95	95	95	95	95	95
		95		95		95		95	
6	95	95	95	95	95	95	95	95	95
		95		95		95		95	

Multi-echelon theory studies this kind of effect and provides models to optimize and synchronize the ordering policy from a system perspective. An example is the concept of the echelon policy. In an echelon ordering policy, the upstream suppliers incorporate all downstream inventories in their ordering decision. This visibility prohibits the demand amplification. Axsäter (2000) provides a good introduction to this policy. The above two examples illustrate that results from single-echelon inventory control can be suboptimal when applied in a multi-echelon context. The examples consider lot sizing and reorder policies.

13.3 Literature Review

Clark and Scarf (1960) are often quoted as the first ones to have studied the problem of multi-echelon inventory control. Recent surveys of relevant contributions can be found in van Houtum (2006), de Kok and Fransoo (2003) and Axsäter (1993). Axsäter (2000) dedicates a chapter on multi-echelon systems in his excellent book "inventory control". Sherbrooke (2004) gives an overview of his pioneering multi-echelon work in the domain of repairables. For repairables we also refer to Muckstadt (2005).

Major steps forward in multi-echelon lot size models have been realized by Crowston *et al.* (1973) and Crowston and Wagner (1973). Under EOQ-like assumptions of constant continuous demand, instantaneous production, zero lead times, infinite planning horizon, fixed setup costs, linear holding costs, etc., they prove that for a multi-stage assembly system, the optimal lot size at each facility is an integer multiple of the lot size at the successor facility. Their proof makes use of the echelon inventory concept first introduced by Clark and Scarf (1960). Even if customer demand is constant, multi-echelon lot sizing problems may be surprisingly complex. Axsäter (2000) shows the optimal solution for a serial system with constant demand can mean order quantities that vary over time. However, Roundy (1985, 1986, 1989) and Muckstadt and Roundy (1993) have shown that it is often possible to use rather simple approximations with a guaranteed cost performance of at most 2% above the optimal costs. They use the so-called "power-of-two" policies, where upstream lot sizes are not only an integer multiple of the downstream lot size, but also a power of two. Federgruen *et al.* (1992), and Federgruen and Zheng (1995, 1993) have further extended the work of Roundy to include joint setup costs and capacitated production/distribution networks. Because of the complexity of the multi-echelon inventory control problem, simulation is increasingly adopted for its investigation. Some recent examples can be found in Köchel and Nieländer (2005), Kaplan *et al.* (2009), and Swaminathan *et al.* (1998).

Cyclic planning approaches are also used in multi-echelon lot sizing. We refer to the recent work of Van den Broecke *et al.* (2003) and to the earlier work of Buzby *et al.* (1999), Campbell (1996), Campbell and Mabert (1991). Multi-echelon cyclic planning is a multi-echelon control policy for which Groenevelt *et al.* (1992) argue that it helps in reducing the well-known bullwhip effect. The bull whip is an illustration of how single echelon control policies can have disturbing effects in a multi-echelon context. Lee *et al.* (1997a) illustrate how demand signal processing, rationing games, order batching and price variations amplify demand in multi-echelon supply chains. More recent work on the bullwhip effect includes that of Holland and Sodhi (2004), who use simulation to analyze the effect of order batching, McCullen and Towill (2002), Chen *et al.* (2000) and Taylor (1999).

Besides the multi-echelon lot sizing and control policies, a third important research area, the one to which this chapter contributes, is that of multi-echelon safety stock optimization. Contrary to the major part of the literature on lot sizing and control policies, demand and/or lead times are now assumed to be stochastic. Safety stock is used to buffer demand and/or lead-time uncertainties and provide reliable service to end customers. In a single-echelon mode, the buffering would typically occur at all inventory points in the network. Multi-echelon models try to decrease the inventory requirements by regrouping the uncertainties in a limited number of inventory points.

De Kok and Fransoo (2003) report that most of the major contributions are rather for uncapacitated systems. Capacitated systems are only available for single-item, single-stage systems, see for example de Kok (1989), serial systems, see for example Tayur (1993), and divergent systems where only the most upstream stage is capacitated, see for example De Kok (2000). Van Houtum (2006) provides an overview of the work that has been done for serial systems, assembly systems and distribution systems. He argues that considering more general production-distribution inventory systems leads to models that are hard to solve to optimality. In practice, however, it is a mixture of convergent and divergent structures that typically prevail. As a result, solution approaches so far have made many simplifying assumptions to make the problem tractable. This chapter focuses on safety stock optimization in rather general networks.

For a detailed review of existing literature on this specific topic, we focus on four sets of related papers. A first set of papers are those by Ettl *et al.* (2000) and Lin *et al.* (2000), both building on Lee and Billington (1993). We will most often refer to Ettl *et al.* (2000) as a major contribution in this set of papers. A second set of papers are those by Graves and Willems (1996, 2000, 2003), Inderfurth and Minner (1998), and Graves (1988). Here also, we will refer to Graves and Willems (2000) as a major contribution in this set of papers. A third set of papers are those by Deuermeyer and Schwarz (1981) and Schwarz *et al.* (1985). We will typically refer to Deuermeyer and Schwarz (1981). A fourth set of papers are those by de Kok and Visschers (1999), Diks and de Kok (1999) and de Kok *et al.* (2005). Here also, we will typically refer to de Kok and Visschers (1999).

While the aim of this chapter is partly to review and analyze the general production–distribution inventory network, not all authors have made the same assumptions. Therefore, to be able to compare and develop solution approaches for the multi-echelon safety stock optimization problem, some fundamental issues need to be addressed. A first issue is the network model being used. Given the network structure, a next issue is the control policy used. Some authors assume a periodic review base-stock policy with a common cycle. This simplifies significantly the characterization of the dependent demand. When discussing lead times, the terminology introduced by Graves and Willems (2003) to differentiate between "nominal" lead times, which assume that goods are available at the input inventory points, and "actual lead times", which include any waiting time for inputs to become available, is used.

Typical network structures are linear networks, distribution networks and assembly networks. Most real-life supply chains combine linear, assembly and distribution networks into "general" production–distribution inventory networks.

Deuermeyer and Schwarz (1981) consider distribution networks. Ettl *et al.* (2000) consider general networks. Graves and Willems (2000) consider spanning tree networks. De Kok and Visschers (1999) consider general networks obeying so-called “weak” and “strong” constraints. The weak constraint requires a unique path from any component to an end-product, i.e. a component cannot be used "in two ways" in an end-product. The small general system shown in Figure 13.5 does not satisfy the weak constraint. This weak constraint is required in the decomposition approach of the assembly network into a series of divergent networks. The strong constraint requires that assembly systems decompose into series systems, i.e. end-products should have different components with the longest cumulative lead-time. This constraint is required for finding optimal base-stock policy.

Ettl *et al.* (2000) assume a continuous-review base-stock (S–1, S) policy with first-come-first-served allocation and back-ordering. Graves and Willems (2000) assume a periodic-review base-stock (S–1, S) policy with common cycle. As we will explain in more detail they assume demand to be bounded, which allows for a 100% service and does not require back-order treatment. Deuermeyer and Schwarz (1981) assume a continuous review (R, Q) policy with reorder point R and fixed lot size Q, first-come-first-served allocation and back-ordering. de Kok and Visschers (1999) assume so-called “synchronized” base-stock (S–1, S) policies. These are periodic-review base-stock (S–1, S) policies with common cycle where the ordering of components is linked to a fair-share pre-allocation of common components with longer lead times. There is no use in ordering more for a shorter lead-time component than the fair share of the longer lead-time component you receive. De Kok and Visschers (1999) find optimal policies under the “balance assumption”, assuming there are no negative allocations in the fair share. Unlike other authors, de Kok and Visschers use echelon stock policies instead of installation stock policies. We refer to Axsäter (2000) for an introduction to echelon stock policies.

In the above major contributions, the nominal lead times are exogenous variables. Ettl *et al.* (2000) assume nominal lead times are stochastic, with a random but known distribution. Graves and Willems (2000), Deuermeyer and Schwarz (1981) and de Kok and Visschers (1999) assume nominal lead times to be deterministic and constant. The actual lead-time is considered endogenous; we first need to characterize service times before we can characterize the actual lead-time. Without loss of generality, external demand is assumed to occur only at stages that have no successors. Ettl *et al.* (2000), Graves and Willems (2000) and de Kok and Visschers (1999) assume external demand to be discrete and random. Graves and Willems (2000) and de Kok and Visschers (1999) assume stationary demand. Ettl *et al.* (2000) allow for non-stationary demand. Deuermeyer and Schwarz (1981) assume a poisson process.

To summarize, perhaps the major assumption of Ettl *et al.* (2000) is the assembly assumption of no concurrent stock-outs. As explained in Graves and Willems (2003) it leads to significant minimal safety stocks in assembly sub-systems and as such hampers the optimization potential. The major assumptions of Graves and

Willems (2000) are the bounded demand and the periodic-review base-stock policy with common cycle. The combination of the two allows for a very elegant conversion of a stochastic demand model into the analysis of a deterministic model with service times as decision variables. The major limitation of the model of Deuermeyer and Schwarz (1981) is probably the network structure. Some major assumptions in de Kok and Visschers (1999) are the weak and the strong constraint for the networks. In this chapter, the main objective is to propose some intelligent and efficient methods for the solution of the safety stock optimization problem in multi-echelon systems. The major assumptions made are the normal approximation for demand during lead-time, the exponential approximation for the back-order service time of a single stock point, the normal approximation for the back-order service time of a set of stock points and the normal approximation of actual lead times.

13.4 System Safety Stock Optimization in *n*-echelon Distribution Systems

In this section we discuss the safety stock optimization in distribution systems. These systems are an intermediate step towards the analysis of the more complex assembly system. The developed approximation for the back-order service time in the central warehouse will be reused there to characterize the back-order service time of the components. Throughout this and next sections we use the following notation:

- $x \sim N(\mu_x, \sigma_x^2)$ indicates that a variable x is normally distributed with average μ_x and variance σ_x^2.
- $x \approx N(\mu_x, \sigma_x^2)$ indicates we have approximated the distribution of a variable x by a normal distribution.
- Φ and φ indicate the cumulative normal probability distribution function and the normal probability distribution function respectively. An index s (Φ_s, φ_s) indicates the standardized version with $\mu = 0$ and $\sigma = 1$.
- $x \sim E(\lambda_x)$ indicates that a variable x is assumed to be exponentially distributed with average μ_x and variance σ_x^2 given by $\lambda_x = \frac{1}{\mu_x} = \frac{1}{\sigma_x}$
- $x \approx E(\lambda_x)$ indicates we have approximated the distribution of a variable x by an exponential distribution.
- $E(\ )$ and $e(\ )$ indicate the cumulative exponential probability distribution function and the exponential probability distribution function respectively.

13.4.1 Introduction of a Two-echelon Distribution System

We consider a one-warehouse *N*-retailer system with the following characteristics:

- Demand to the retailers is normally distributed, that is $d_i \sim N\left(\mu_{d_i}, \sigma_{d_i}^2\right)$ where the index *i* refers to retailer *i*.
- The nominal replenishment lead times for the retailers are normally distributed, that is $L_i^0 \sim N\left(\mu_{L_i^0}, \sigma_{L_i^0}^2\right)$. The nominal replenishment lead times assume that goods are available in the warehouse. Nominal lead times carry an index 0.
- As stock-outs may occur in the warehouse, actual replenishment lead-time will differ from the nominal lead-time. The actual lead-time includes a service time for back-orders in the warehouse. In what follows we approximate the back-order service time at the warehouse by an exponential distribution. We will denote it as $S_c^B \approx E\left(\lambda_{S_c^B}\right)$. The index c indicates that the warehouse is central to the distribution system. We approximate the actual replenishment lead-time of the retailers by a normal distribution and denote it as $L_i \approx N\left(\mu_{L_i}, \sigma_{L_i}^2\right)$.
- For normally distributed demand and normally distributed actual lead times, the demand during the actual lead-time is normally distributed, $D_i \sim N\left(\mu_{D_i}, \sigma_{D_i}^2\right)$, with $\mu_{D_i} = \mu_{d_i}\mu_{L_i}$ and $\sigma_{D_i}^2 = \mu_{L_i}\sigma_{d_i}^2 + \mu_{d_i}^2\sigma_{L_i}^2$
- We calculate safety stocks in the retailers on the basis of a volume fill rate, that is $SS_i = k_i\sigma_{D_i}$ with k_i determined from a target volume fill rate f_i through inversion of $f_i = 1 - \frac{1}{Q_i}\left[G\left(R_i, \mu_{D_i}, \sigma_{D_i}\right) - G\left(R_i + Q_i, \mu_{D_i}, \sigma_{D_i}\right)\right]$, where R_i is the reorder point $R_i = \mu_{D_i} + SS_i$, Q_i is a fixed lot and the help function *G* is defined as $G\left(r, \mu_D, \sigma_D\right) = \left(\mu_D - r\right)\left[1 - \Phi\left(r; \mu_D; \sigma_D^2\right)\right] + \sigma_D^2\varphi\left(r; \mu_D; \sigma_D^2\right)$. We refer to Axsäter (2000) for the derivation of this expression. This inversion has been implemented numerically by the authors and can be used as a function in Microsoft Excel®.
- The reorder process at the retailers is based on a $\left(R_i, Q_i\right)$ policy where R_i is a fixed reorder point and Q_i is a fixed lot size.
- Retailer orders to the warehouse are delivered following the first-in-first-out rule and in full amount. Any retailer order that cannot be delivered in full amount is back-ordered until sufficient goods are available.
- In the case of identical retailers, the central demand can be characterized as a binomial distribution with the probability for success $p = \frac{\mu_{d_r}}{Q_r}$ and the number of trials *n* equal to the number of retailers *N*. In general the number of orders *o*

to the warehouse is binomially distributed with average $\mu_{o_c} = \frac{\mu_{d_r}}{Q_r} N$ and standard deviation $\sigma_{o_c} = \sqrt{N \frac{\mu_{d_r}}{Q_r}\left(1 - \frac{\mu_{d_r}}{Q_r}\right)}$. The demand distribution is derived by multiplying σ_{o_c} by the order size Q. We use the resulting average and standard deviation for a normal approximation of the central demand, that is $d_c \approx N\left(\mu_{d_c}, \sigma^2_{d_c}\right)$ with $\mu_{d_c} = N\mu_{d_r}$ and $\sigma_{d_c} = Q_r \sqrt{N \frac{\mu_{d_r}}{Q_r}\left(1 - \frac{\mu_{d_r}}{Q_r}\right)}$. This follows a reasoning of Lee *et al.* (1997b).

- In the case of non-identical retailers, we assume the demand to the warehouse is normally distributed with average and variance given as $\mu_{d_c} = \sum_i \mu_{d_i}$ and $\sigma^2_{d_c} = \sum_i \sigma^2_{d_i}$. This assumption ignores the demand amplification from the reorder policies. In practical applications usually central demand can be observed and its average and variance can be measured.
- We assume the replenishment source of the warehouse to have an infinite stock. As such the nominal and actual replenishment lead-time of the warehouse are equal. We assume it to be normally distributed and denote it as $L_c \sim N\left(\mu_{L_c}, \sigma^2_{L_c}\right)$.
- The demand during the lead-time for the warehouse is normally distributed and denoted as $D_c \sim N\left(\mu_{D_c}, \sigma^2_{D_c}\right)$, with $\mu_{D_c} = \mu_{d_c}\mu_{L_c}$ and $\sigma^2_{D_c} = \mu_{L_c}\sigma^2_{d_c} + \mu^2_{d_c}\sigma^2_{L_c}$.
- In full analogy to the retailers, we calculate the warehouse safety stock on the basis of a volume fill rate f_c, that is $SS_c = k_c\sigma_{D_c}$ with k_c determined through inversion of $f_c = 1 - \frac{1}{Q_c}\left[G\left(R_c, \mu_{D_c}, \sigma_{D_c}\right) - G\left(R_c + Q_c, \mu_{D_c}, \sigma_{D_c}\right)\right]$, where R_c is the reorder point $R_c = \mu_{D_c} + SS_c$ and Q_c is a fixed lot size, and the function G is as defined above.
- The reorder process at the warehouse is also based on an $\left(R_c, Q_c\right)$ policy.

Figure 13.8 illustrates the system and its parameters.

Example System 1

We analyze the above mathematical models for an example of a 10-identical retailer distribution system. The demand to the retailers is normally distributed with an average of 100 units per day and a standard deviation of 30 units per day. The relatively low coefficient of demand variation of 30% highlights the fast-moving character. The average nominal replenishment lead-time is 5 days and its standard deviation is 1 day. The retailer lot size for reordering is 500 units. This implies that they reorder on average every 5 days. We assume the warehouse is being replenished with an average lead-time of 30 days and a standard deviation of 6 days. The DC lot size for reordering is 5000 units. This implies that the warehouse also reorders on average every 5 days. Average and variance of the central demand are calculated using the binomial expressions derived above. A common situation for companies is to have the same target service level at both the retailers and the central warehouse. We assume the target service to be 95%. Table 13.2 summarizes the distribution environment.

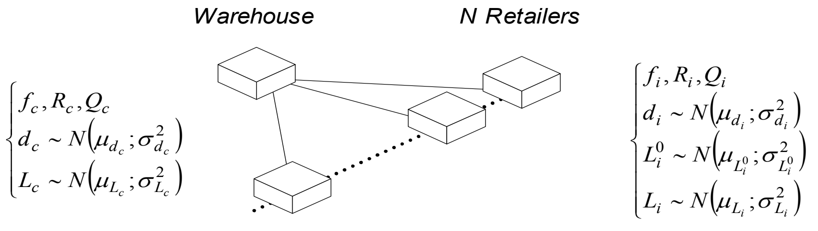

Warehouse
N Retailers
f_c, R_c, Q_c
$d_c \sim N(\mu_{d_c}; \sigma^2_{d_c})$
$L_c \sim N(\mu_{L_c}; \sigma^2_{L_c})$
f_i, R_i, Q_i
$d_i \sim N(\mu_{d_i}; \sigma^2_{d_i})$
$L_i^0 \sim N(\mu_{L_i^0}; \sigma^2_{L_i^0})$
$L_i \sim N(\mu_{L_i}; \sigma^2_{L_i})$

Fig. 13.8 A two-echelon distribution system

Table 13.2 Example two-echelon 10-identical retailer distribution system: parameters

Retailer		Warehouse	
N	10		
f_r	95%	f_r	95%
Q_r	500	Q_r	5000
μ_{dr}	100	μ_{dc}	1000
σ_{dr}	30	σ_{dc}	632
μ_{Lr0}	5	μ_{Lc}	30
σ_{Lr0}	1	σ_{Lc}	6

13.4.2 Characterization of the Back-order Service Time in the Central Warehouse

The service time for back-orders in the warehouse, denoted by S_c^B, is the time between the registration of the back-order and its allocation against physical inventory. It is the time to "resolve" back-orders. Figure 13.9 shows the back-order service time as measured in a discrete-event simulation of the example system introduced above. The four graphs are for a central service level 85%, 65%, 45%

and 25% respectively. The left y-axis shows the number of back-order occurrences in the simulation (dots). The x-axis shows the number of periods it took to resolve the back-orders. Reducing the central service level increases the number of back-orders and their average time to be resolved. The right y-axis shows an exponential approximation $S_c^B \approx E\left(\frac{1}{\mu_{S_c^B}}\right)$, with the average $\mu_{S_c^B}$ calculated using the result of Lemma 13.1 below.

Lemma 13.1: *Under the assumptions outlined above, the expected value for the service time for back-orders can be calculated as*

$$\mu_{S_c^B} = \frac{\int_{-\infty}^{+\infty} \frac{G\left(t,\mu_{L_c},\sigma_{L_c}\right)}{1-\Phi\left(t;\mu_{L_c};\sigma_{L_c}^2\right)}.\left[1-\Phi\left(R_c;t.\mu_{d_c};t.\sigma_{d_c}^2\right)\right].\left[1-\Phi\left(t;\mu_{L_c};\sigma_{L_c}^2\right)\right].dt}{\int_{-\infty}^{+\infty}\left[1-\Phi\left(R_c;t.\mu_{d_c};t.\sigma_{d_c}^2\right)\right].\left[1-\Phi\left(t;\mu_{L_c};\sigma_{L_c}^2\right)\right].dt}. \quad (13.1)$$

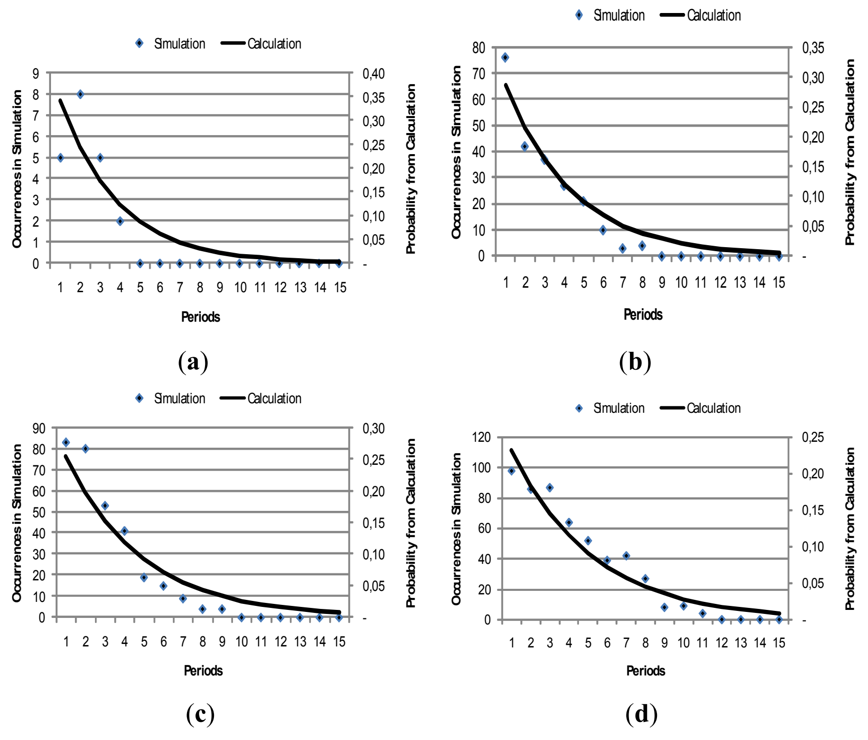


Fig. 13.9 Back-order service time in the warehouse: exponential approximation and histogram from simulation – for warehouse service levels of 85% (**a**), 65% (**b**), 45% (c) and 25% (**d**)

For a proof of Lemma 13.1 we refer to Desmet *et al.* (2009a). It basically considers all possible time moments *t* in a reorder cycle and weighs the waiting time at time moment *t* with the probability for a stock-out to occur at time moment *t.* A numerical evaluation of Eq. 13.1 assumes a numerical expression of the normal distribution, cumulative normal distribution and its standard variants and a numerical integration. This function is implemented by the authors and is available as an add-in in Microsoft Excel®.

13.4.3 Characterization of the Actual Retailer Replenishment Lead Time

Using the above characterization of the service time for back-orders, we can derive an approximation for the actual retailer replenishment lead-time.

Lemma 13.2: *Under the above assumptions the actual retailer replenishment lead-time L_i can be approximated by a normal distribution with average and variance as indicated in the following expression*:

$$L_i \approx N\left(\mu_{L_i^0} + \mu_{S_c^B}.(1-f_c);\sigma_{L_i^0}^2 + \sigma_{S_c^B}^2.(1-f_c)\right). \qquad (13.2)$$

We again refer to Desmet *et al.* (2009a) for a proof of Lemma 13.2. In essence we approximate the order line service level in the central warehouse via the volume fill rate f_c. The actual lead-time is then the probability-weighted sum of the nominal lead-time distribution and the convolution of the nominal lead-time distribution and the back-order service-time distribution. The respective probabilities are the probability for a hit and the probability for a stock-out in the central warehouse. We apply a normal approximation with as average and variance the probability weighted averages and variances. This leads to Eq.13.2.

If from Eq. 13.1: $\mu_{S_c^B}\left(R\left(f_c,Q_c,\mu_{d_c},\sigma_{d_c},\mu_{L_c},\sigma_{L_c}\right),\mu_{d_c},\sigma_{d_c},\mu_{L_c},\sigma_{L_c}\right)$ then from Eq. 13.2 we have $\mu_{L_i}\left[\mu_{L_i^0},1-f_c,\mu_{S_c^B}\left[R\left(f_c,Q_c,\mu_{d_c},\sigma_{d_c},\mu_{L_c},\sigma_{L_c}\right),\mu_{d_c},\sigma_{d_c},\mu_{L_c},\sigma_{L_c}\right]\right]$ and in analogy $\sigma_{L_i}^2\left[\sigma_{L_i^0}^2,1-f_c,\sigma_{S_c^B}^2\left[R\left(f_c,Q_c,\mu_{d_c},\sigma_{d_c},\mu_{L_c},\sigma_{L_c}\right),\mu_{d_c},\sigma_{d_c},\mu_{L_c},\sigma_{L_c}\right]\right]$ or the actual replenishment lead-time for the retailers depends on their nominal replenishment lead-time and all parameters influencing the warehouse stock.

Example System 1 (Continued)

The following graphs (Figure 13.10) show the actual and the nominal retailer replenishment lead-time for varying warehouse service levels from a discrete-event simulation for the above small example system. The exponential back-order simulation for the above small example system. The exponential back-order distribution increasingly skews the nominal lead-time distribution as the central service level is reduced. The actual lead-time, shown in dots, is what we approximate using a normal distribution through Lemma 13.2.

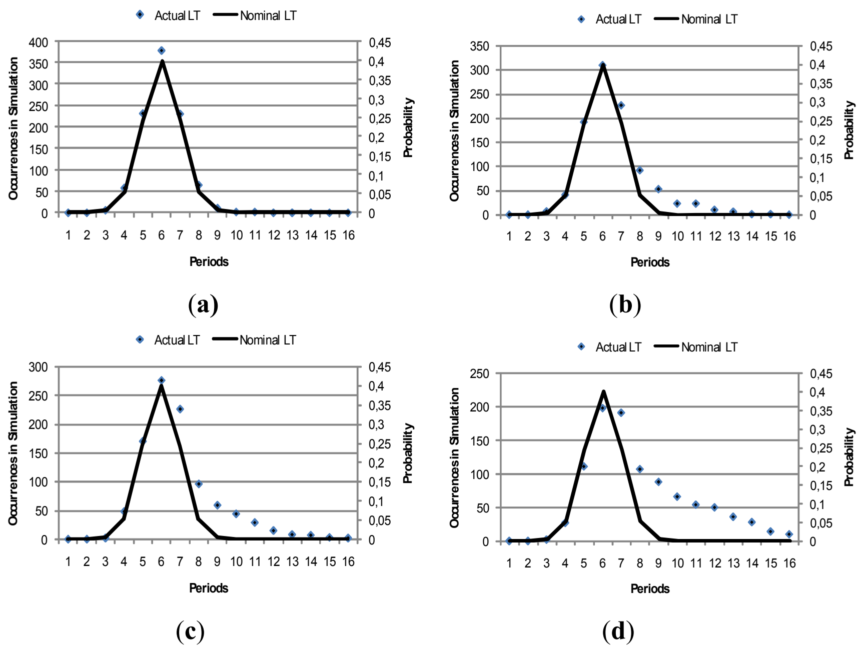


Fig. 13.10 Actual and nominal retailer replenishment lead-time: histogram from simulation – for warehouse service levels of 85% (**a**), 65% (**b**), 45% (**c**) and 25% (**d**)

The following two graphs (Figure 13.11) show the calculation and the simulation results for the average and standard deviation of the actual retailer lead-time for the example system. The approximation for the average is conservative down to a service level of 35%. The proposed approximation for the standard deviation is rather conservative over the full line.

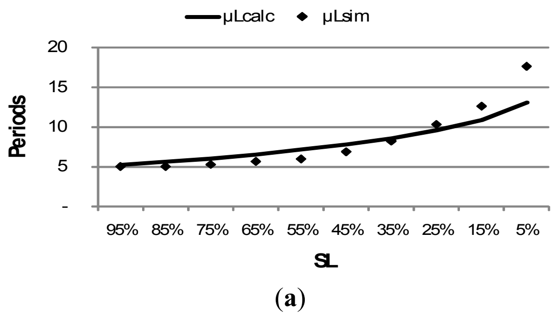
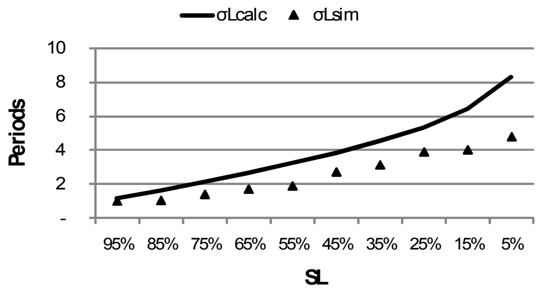


Fig. 13.11 Actual retailer replenishment lead-time: calculation versus simulation results for average (**a**) and standard deviation (**b**)



(b)

Fig. 13.11 (continued)

We can now apply the above results in optimizing the system safety stock.

13.4.4 System Safety Stock Optimization in a Two-echelon Distribution System

As indicated above we calculate safety stocks on the basis of a volume fill rate, that is $SS_i = k_i \sigma_{D_i}$ with k_i found from a target volume fill rate f_i through inversion of $f_i = 1 - \frac{1}{Q_i}\left[G\left(R_i, \mu_{D_i}, \sigma_{D_i}\right) - G\left(R_i + Q_i, \mu_{D_i}, \sigma_{D_i}\right)\right]$, with R_i the reorder point $R_i = \mu_{D_i} + SS_i$, Q_i a fixed lot, the help function G defined as $G(r, \mu_D, \sigma_D) = (\mu_D - r)\left[1 - \Phi\left(r; \mu_D; \sigma_D^2\right)\right] + \sigma_D^2 \varphi\left(r; \mu_D; \sigma_D^2\right)$ where $\mu_{D_i} = \mu_{d_i}\mu_{L_i}$ and $\sigma_{D_i} = \sqrt{\mu_{L_i}\sigma_{d_i}^2 + \mu_{d_i}^2 \sigma_{L_i}^2}$.

Optimizing the system safety stock is probably most comprehensive if we rewrite the system safety stock as follows:

$$SS_{tot} = k_c\left(f_c, Q_c, \mu_{d_c}, \sigma_{d_c}, \mu_{L_c}, \sigma_{L_c}\right)\sqrt{\mu_{L_c}\sigma_{d_c}^2 + \mu_{d_c}^2\sigma_{L_c}^2} + \sum_{i=1}^{N} k_i\left(f_i, Q_i, \mu_{d_i}, \sigma_{d_i}, \mu_{L_i}, \sigma_{L_i}\right)\sqrt{\mu_{L_i}\sigma_{d_i}^2 + \mu_{d_i}^2\sigma_{L_i}^2} \tag{13.3}$$

with $\mu_{L_i}\left[\mu_{L_i^0}, 1 - f_c, \mu_{S_c^B}\left[R\left(f_c, Q_c, \mu_{d_c}, \sigma_{d_c}, \mu_{L_c}, \sigma_{L_c}\right), \mu_{d_c}, \sigma_{d_c}, \mu_{L_c}, \sigma_{L_c}\right]\right]$ and $\sigma_{L_i}^2\left[\sigma_{L_i^0}^2, 1 - f_c, \sigma_{S_c^B}^2\left[R\left(f_c, Q_c, \mu_{d_c}, \sigma_{d_c}, \mu_{L_c}, \sigma_{L_c}\right), \mu_{d_c}, \sigma_{d_c}, \mu_{L_c}, \sigma_{L_c}\right]\right]$ as derived above through Eqs. 13.1 and 13.2.

The warehouse service level f_c is the single decision variable. Lot sizes, demand distributions, nominal lead-time distributions and retailer fill rates f_i are inputs to the model. Reducing f_c:

1. reduces the warehouse safety stock (in reducing k_c);
2. increases the back-order service time,
   – as such the actual replenishment lead times for the retailers and
   – as such their required safety stock for a the given fill rate f_i.

For problems with one decision variable many optimization techniques are available. So far we have used a one-dimensional grid search and used a visual representation to increase insight in the sensitivity into the solution.

Example System 1 (Continued)

Figure 13.12 shows the results of the grid search for the small example system. The central safety stock becomes zero for a fill rate of 64%. This boundary of 64% turns out to be optimal. The decrease in warehouse inventory is faster than the corresponding increase in retailer inventory. It shows that in the positive safety stock zone the proposed approximations are conservative. This implies that we even overstate the required increase of retailer safety stocks. As we will show in Sect. 13.6 it is not uncommon to find significant central safety stocks in real life distribution systems. In a single-echelon mode all stock points, including the central warehouse, typically have the same service target. For this example system this would be 95%. Furthermore, Figure 13.12 shows that the benefits of multi-echelon safety stock optimization can be substantial.

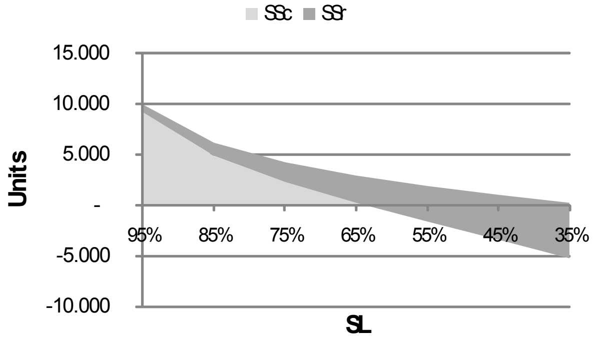


Fig. 13.12 System safety stock optimization in a two-echelon 10-identical retailer distribution system

For the small example system the total safety stock can be reduced from 9896 to 2898 units. That is a remarkable reduction of 70%. The result also has important consequences for the central and the decentral warehouses. The safety stock in the central warehouse disappears. The required safety stock for the retailers increases from 756 to 2671 units, which is an increase by a factor 3.53. There may be physical limitations in space when trying to implement this solution, which means that the implementation also requires some type of central control. When making this type of consideration Graves and Willems (2000) rightly talk about "strategic" safety stock positioning in multi-echelon supply chains.

Taking into account the added value of the product will act as a counter force to shifting inventories downstream. However, the literature in general indicates it is more favourable to shift safety stocks closer to the customer. This conclusion is far more stable than at least many practitioners are willing to accept.

Extension to *n*-echelon Distribution Systems

The approach shown for the two-echelon distribution system can easily be extended to *n*-echelon distribution systems. Stages that have no predecessor can assume the actual lead-time is equal to the nominal lead-time. For these stages we can then characterize the service time for back-orders using Eq. 13.1. The stages that are dependent on these stages can use this back-order service to calculate their actual lead-time using Eq. 13.2. Given their actual lead times they can calculate a back-order service time. We will elaborate on this type of recursive algorithms in more detail for generic networks in Sect. 13.5. We hope for now the above description is sufficiently clear to indicate that indeed the approach can be extended to *n*-echelon distribution networks.

13.5 System Safety Stock Optimization in *n*-echelon Assembly Systems

In this section we discuss the safety stock optimization in assembly systems. The developed approach reuses the exponential approximation for the back-order service time in the central warehouse from Sect. 13.4 on distribution systems.

13.5.1 Introduction of a Two-echelon Assembly System

In analogy to the one-warehouse *N*-retailer system of Sect. 13.4, we now consider an assembly system with *N* components assembled to one assembly with the following characteristics:

- Demand for assemblies is normally distributed, that is $d_a \sim N\left(\mu_{d_a}, \sigma_{d_a}^2\right)$. The index "*a*" refers to assembly.

- The nominal lead-time for the assembly operation is normally distributed, that is $L_a^0 \sim N\left(\mu_{L_a^0}, \sigma_{L_a^0}^2\right)$. The nominal lead-time assumes all inputs are available to start the assembly operation. The index 0 indicates that assumption.
- As the *N* components, required for the assembly operation, experience stock-outs, the actual lead-time is different from the nominal lead-time. The actual lead-time includes an incoming service time which is the combined effect of the components experiencing stock-outs. In what follows we will approximate the incoming service time by a normal distribution and denote it by $S_a^I \approx N\left(\mu_{S_a^I}, \sigma_{S_a^I}^2\right)$. The index *I* indicates that it is an incoming service time. The index *a* indicates it is the assembly operation. We will also approximate the actual lead-time of the assembly operation by a normal distribution and denote it as $L_a \approx N\left(\mu_{L_a}, \sigma_{L_a}^2\right)$ (without the index 0). The actual lead-time is to be used in the calculation of the safety stock of assemblies.
- For normally distributed demand and a normal approximation for the actual lead-time, the demand during the actual assembly lead-time can be approximated by a normal distribution, that is $D_a \approx N\left(\mu_{D_a}, \sigma_{D_a}^2\right)$, with $\mu_{D_a} = \mu_{d_a}\mu_{L_a}$ and $\sigma_{D_a}^2 = \mu_{L_a}\sigma_{d_a}^2 + \mu_{d_a}^2\sigma_{L_a}^2$.
- We calculate the safety stock for assemblies on the basis of a volume fill rate, that is $SS_a = k_a\sigma_{D_a}$ with k_a found from a target volume fill rate f_a through inversion of $f_a = 1 - \frac{1}{Q_a}\left[G\left(R_a, \mu_{D_a}, \sigma_{D_a}\right) - G\left(R_a + Q_a, \mu_{D_a}, \sigma_{D_a}\right)\right]$, with R_a the reorder point $R_a = \mu_{D_a} + SS_a$, Q_a a fixed lot size and the help function $G\left(r, \mu_D, \sigma_D\right) = \left(\mu_D - r\right)\left[1 - \Phi\left(r; \mu_D; \sigma_D^2\right)\right] + \sigma_D^2\varphi\left(r; \mu_D; \sigma_D^2\right)$. The reorder process at the assembly operation is based on a $\left(R_a, Q_a\right)$ policy with $R_a = \mu_{D_a} + SS_a$ and Q_a a fixed lot size.
- An order for components is to be delivered in full. In case one or more components cannot be delivered these components are back-ordered for the full amount. The assembly process can only start operation when all components are available.
- In analogy to the identical retailer case given above, the demand for components can be characterized by a binomial distribution. The probability for success *p* is $\frac{\mu_{d_a}}{Q_a}$. The number of trials n_a now equals 1. In general the number of orders o_i for a component *i* is binomially distributed with $\mu_{o_i} = \frac{\mu_{d_a}}{Q_a} n_a$ and

$\sigma_{o_i} = \sqrt{n_a \frac{\mu_{d_a}}{Q_a}\left(1-\frac{\mu_{d_a}}{Q_a}\right)}$. The demand distribution is derived in multiplying with the order size Q_a. We will use the resulting average and standard deviation for a normal approximation of the demand for component *I*, that is $d_i \sim N\left(\mu_{d_i}, \sigma_{d_i}^2\right)$ with $\mu_{d_i} = \mu_{d_i}$ and $\sigma_{d_i} = Q_a\sqrt{\frac{\mu_{d_a}}{Q_a}\left(1-\frac{\mu_{d_a}}{Q_a}\right)}$.

- We assume the suppliers of the components have an infinite stock. As such the nominal and actual replenishment lead-time for the components are equal. We assume them to be normally distributed and denote them as $L_i \sim N\left(\mu_{L_i}, \sigma_{L_i}^2\right)$.
- The demand during lead-time for the components is normally distributed and denoted as $D_i \sim N\left(\mu_{D_i}, \sigma_{D_i}^2\right)$, with $\mu_{D_i} = \mu_{d_i}\mu_{L_i}$ and $\sigma_{D_i}^2 = \mu_{L_i}\sigma_{d_i}^2 + \mu_{d_i}^2\sigma_{L_i}^2$.
- In full analogy to the assembly, we calculate the component safety stocks on the basis of a volume fill rate f_i, that is $SS_i = k_i\sigma_{D_i}$ with k_i found through inversion of $f_i = 1 - \frac{1}{Q_i}\left[G\left(R_i, \mu_{D_i}, \sigma_{D_i}\right) - G\left(R_i + Q_i, \mu_{D_i}, \sigma_{D_i}\right)\right]$, with R_i the reorder point $R_i = \mu_{D_i} + SS_i$, Q_i a fixed lot size and the help function $G\left(r, \mu_D, \sigma_D\right) = \left(\mu_D - r\right)\left[1 - \Phi\left(r; \mu_D; \sigma_D^2\right)\right] + \sigma_D^2\varphi\left(r; \mu_D; \sigma_D^2\right)$.
- The reorder process for components is also based on a $\left(R_i, Q_i\right)$ policy with $R_i = \mu_{D_i} + SS_i$ and Q_i a fixed lot size for component *i*.

Figure 13.13 illustrates the system and its parameters.

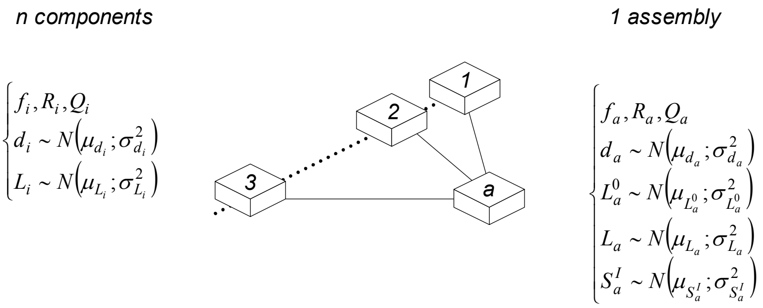

n components
1 assembly
f_i, R_i, Q_i
$d_i \sim N\left(\mu_{d_i}; \sigma_{d_i}^2\right)$
$L_i \sim N\left(\mu_{L_i}; \sigma_{L_i}^2\right)$
1
2
3
a
f_a, R_a, Q_a
$d_a \sim N\left(\mu_{d_a}; \sigma_{d_a}^2\right)$
$L_a^0 \sim N\left(\mu_{L_a^0}; \sigma_{L_a^0}^2\right)$
$L_a \sim N\left(\mu_{L_a}; \sigma_{L_a}^2\right)$
$S_a^I \sim N\left(\mu_{S_a^I}; \sigma_{S_a^I}^2\right)$

Fig. 13.13 A two-echelon assembly system

Example System 2

We will compare the proposed analytical models with the results of a discrete-event simulation for an example 10-identical component assembly system. We assume demand for assemblies to be relatively stable with an average of 100 pieces (pc) per period and a standard deviation of 30 pc. The nominal assembly time is assumed to be on average five periods with a standard deviation of one period. The assembly lot size is 500 pc, which reflects a traditional batch driven production environment. The target fill rate for assemblies is assumed to be 95%. Supply lead times for components are assumed to be on average 30 periods, with a standard deviation of six periods. If the periods were days, this could reflect sourcing in Asia. Demand for components is assumed to be normal with average of 100 and a standard deviation of 200, calculated using the binomial expression $\sigma_{d_i} = Q_a \sqrt{\frac{\mu_{d_a}}{Q_a}\left(1 - \frac{\mu_{d_a}}{Q_a}\right)}$ derived above. Component supply is done per 1500 pc. The example assembly environment is summarized in Table 13.3. The index *u* refers to upstream, the index *a* to assembly.

Table 13.3 Example two-echelon 10-identical component assembly system: parameters

Components		Assembly	
n	10		
f_u	95%	f_a	95%
Q_u	1500	Q_a	500
μ_{du}	100	μ_{da}	100
σ_{du}	200	σ_{da}	30
μ_{Lu}	30	μ_{La}	5
σ_{Lu}	6	σ_{La}	1

13.5.2 Characterization of the Back-order Service Time for a Subset of Components

Consider a subset of components $A \subseteq \{1,...,N\}$, with $\# A = m$, experiencing a concurrent stock-out. We define the back-order service time of the subset A, that is S_A^B, as the time it takes for all back-orders in A to be resolved. The following lemma derives a normal approximation for S_A^B.

Lemma 13.3: *Under the above assumptions the back-order service time of a subset of components A can be approximated by a normal distribution, that is* $S_A^B \approx N\left(\mu_{S_A^B}, \sigma^2_{S_A^B}\right)$, *with average and variance following from numerical integration of:*

$$\mu_{S_A^B} = m\int_{-\infty}^{+\infty} xe\left(x, \frac{1}{\max\left(\mu_{S_j^B} \middle| j \in A\right)}\right) E\left(x, \frac{1}{\max\left(\mu_{S_j^B} \middle| j \in A\right)}\right)^{m-1} dx \tag{13.4}$$

$$\sigma_{S_A^B}^2 = m\int_{-\infty}^{+\infty} \left(x - \mu_{S_A^B}\right)^2 e\left(x, \frac{1}{\max\left(\mu_{S_j^B} \middle| j \in A\right)}\right) E\left(x, \frac{1}{\max\left(\mu_{S_j^B} \middle| j \in A\right)}\right)^{m-1} dx \tag{13.5}$$

where e and E are the exponential probability and cumulative probability distribution function respectively.

For a proof of Lemma 13.3 we refer to Desmet *et al.* (2009b). The approximation in essence assumes that all components behave according to the maximal back-order service-time distribution. The above expressions calculate the expected value and the variance of the maximum distribution of *m* identically exponentially distributed variables.

13.5.3 Characterization of the Incoming Service Time to the Assembly

The incoming service time to the assembly operation then follows in considering all possible subsets *A* and calculating the probability weighted sum of the resulting back-order service time distributions. The weights are the probabilities that exactly the subset *A* experiences a concurrent stock-out. If we again approximate the order line fill rate of a component *i* with the fill rate f_i that probability is given by $\prod_{j \notin A} f_j \prod_{j \in A} (1 - f_j)$. If we again approximate the result by a normal distribution this leads to the result of Lemma 13.4. For a detailed proof we refer to Desmet *et al.* (2009b).

Lemma 13.4: *Under the above assumptions the incoming service time to the assembly operation can be approximated by a normal distribution, that is* $S_a^I \approx N\left(\mu_{S_a^I}, \sigma_{S_a^I}^2\right)$, *with average and variance as shown in the following expression*:

$$S_a^I \approx N\left(\sum_A \prod_{j \notin A} f_j \prod_{j \in A} (1 - f_j)\mu_{S_A^B}; \sum_A \prod_{j \notin A} f_j \prod_{j \in A} (1 - f_j)\sigma_{S_A^B}^2\right) \tag{13.6}$$

with the sum is taken over all possible subsets *A* of the components.

13.5.4 Characterization of the Actual Assembly Lead Time

Given a normal distribution for the nominal lead-time and a normal approximation for the incoming service time, the actual assembly lead-time can be easily approximated by a normal distribution as shown in the following equation:

$$L_a \approx N\left(\mu_{L_a^0} + \mu_{S_a^I}, \sigma_{L_a^0}^2 + \sigma_{S_a^I}^2\right) \tag{13.7}$$

Figure 13.14 and Figure 13.15 show the calculation and simulation results for the actual assembly lead-time in the example system.

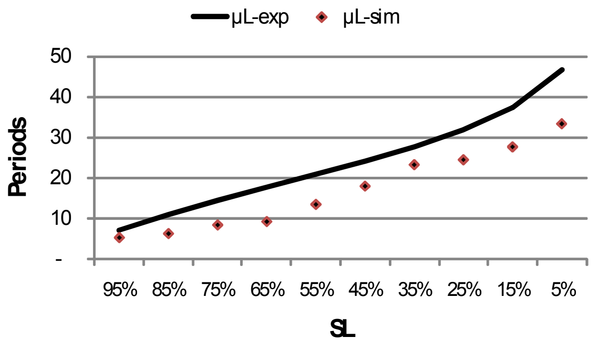


Fig. 13.14 Actual assembly lead-time: calculation *versus* simulation results for the average

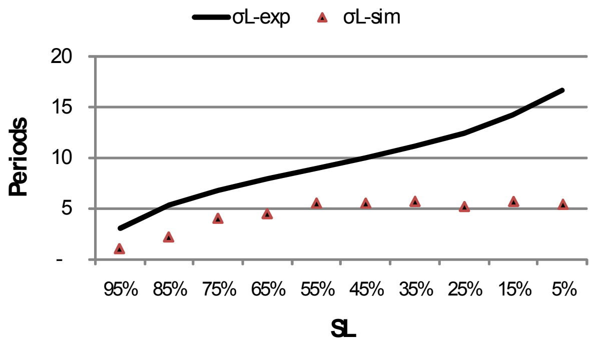


Fig. 13.15 Actual assembly lead-time: calculation *versus* simulation results for the standard deviation

Figure 13.14 shows that the developed approximation for the average of the actual assembly lead-time is conservative. The approximation is best for the high service level of 95% and for a low service level of 35%. The simulation results show an infliction point near 35%. Figure 13.16 can explain this. It shows the expected service level for the components from the calculation *versus* the measured service level in the discrete-event simulation. As introduced above, the actual demand distribution is a binomial and we have approximated it by a normal distribution. This leads to an underestimation of high service levels and an overestimation of low service levels. The service level graph shows the same infliction point near 35%. Figure 13.15 shows the approximation for the standard deviation of the actual lead-time is also conservative. The simulation results seem to stabilize around a standard deviation of five periods. The proposed model expects a further increase.

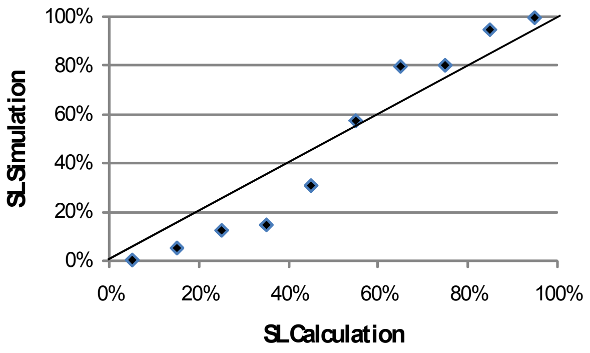


Fig. 13.16 Service level of components: calculation *versus* simulation results

As for the distribution system, we will now analyze the impact on the system safety stock using the developed approximations.

13.5.5 System Safety Stock Optimization in a Two-echelon Assembly System

As introduced above, we calculate safety stocks on the basis of a volume fill rate, that is $SS_a = k_a \sigma_{D_a}$ with k_a found from a target volume fill rate f_a through inversion of $f_a = 1 - \frac{1}{Q_a}\left[G\left(R_a, \mu_{D_a}, \sigma_{D_a}\right) - G\left(R_a + Q_a, \mu_{D_a}, \sigma_{D_a}\right)\right]$, with R_a the reorder point $R_a = \mu_{D_a} + SS_a$, Q_a a fixed lot size and the help function G as defined as $G(r, \mu_D, \sigma_D) = (\mu_D - r)\left[1 - \Phi\left(r; \mu_D; \sigma_D^2\right)\right] + \sigma_D^2 \varphi\left(r; \mu_D; \sigma_D^2\right)$. Equation 13.8 shows the system safety stock.

$$SS_{tot} = \sum_{i=1}^{N} k_i\left(f_i, Q_i, \mu_{d_i}, \sigma_{d_i}, \mu_{L_i}, \sigma_{L_i}\right)\sqrt{\mu_{L_i}\sigma_{d_i}^2 + \mu_{d_i}^2\sigma_{L_i}^2}$$
$$+k_a\left(f_a, Q_a, \mu_{d_a}, \sigma_{d_a}, \mu_{L_a}, \sigma_{L_a}\right)\sqrt{\mu_{L_a}\sigma_{d_a}^2 + \mu_{d_a}^2\sigma_{L_a}^2}. \quad (13.8)$$

μ_{L_a} and σ_{L_a} are found from Eq. 13.7 and $\mu_{S_a^I}$ and $\sigma_{S_a^I}$ from Eq. 13.6. The component service levels f_i are the decision variables. Figure 13.17 shows the results of a grid search for our example 10-identical-component system. The component safety stocks become zero for a fill rate of 71%. This boundary of 71% turns out to be the optimum. The decrease in component safety stocks is faster than the corresponding increase in assembly safety stock. Figures 13.14 and 13.15 show that in the positive safety stock zone the proposed approximations are conservative. This implies we even overstate the required assembly safety stock.

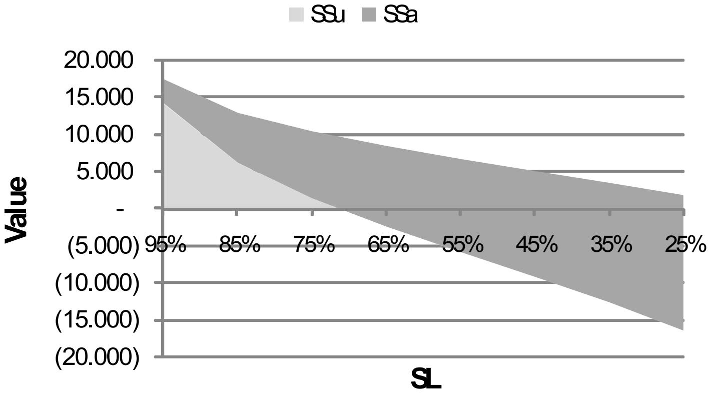


Fig. 13.17 System safety stock optimization in a two-echelon 10-identical component assembly system

It is not uncommon to find significant component safety stocks in assembly systems. In a single-echelon mode all stock points, including the components, typically have the same service target. In our example system this would be 95%. Figure 13.17 illustrates the benefits of multi-echelon safety stock optimization can be substantial. For the example system the safety stock can be reduced from 17.493 value units to 9.749 value units. That is a reduction of 44%. Given a cost of one per component Figure 13.17 assumes a cost of 10 for an assembly.

The result has important consequences on the stocking requirements for components and assemblies. All component safety stock disappears and the assembly safety stock needs to be increased from 3.247 units to 9.749 units. That is a factor 3. There may be physical limitations to accommodate this type of increase. This again justifies the term "strategic" safety stock positioning used by Graves and Willems (2000). Taking into account the added value of the product will act as a counter force to shifting inventories downstream. However, as with distribution systems, the literature in general indicates it is more favorable to shift safety stocks closer to the customer.

13.5.6 Distribution System as Special Case of the Assembly System

Consider our two-echelon distribution system introduced in Sect. 13.4 We can decompose this system into N assembly systems where each retailer, has exactly one input, the finished product from the central warehouse, which we'll label with u, from upstream. We label the assemblies with a. This is illustrated in Figure 13.18.

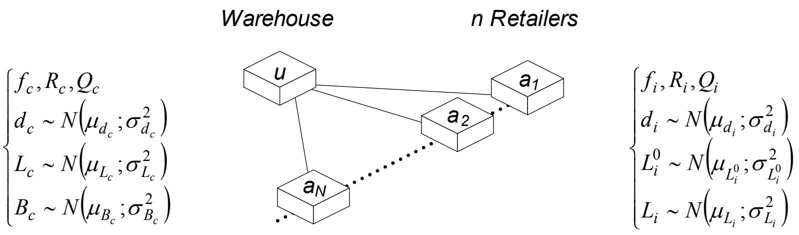

Warehouse
n Retailers
$\begin{cases} f_c, R_c, Q_c \\ d_c \sim N(\mu_{d_c}; \sigma^2_{d_c}) \\ L_c \sim N(\mu_{L_c}; \sigma^2_{L_c}) \\ B_c \sim N(\mu_{B_c}; \sigma^2_{B_c}) \end{cases}$
u
a1
a2
aN
$\begin{cases} f_i, R_i, Q_i \\ d_i \sim N(\mu_{d_i}; \sigma^2_{d_i}) \\ L_i^0 \sim N(\mu_{L_i^0}; \sigma^2_{L_i^0}) \\ L_i \sim N(\mu_{L_i}; \sigma^2_{L_i}) \end{cases}$

Fig. 13.18 Distribution system as a special case of the assembly system

If we consider the assembly system $\{u, a_i\}$, the possible subsets of components A needed for Eq. 13.6 are now $A_0 = \{\ \}$ and $A_1 = \{u\}$.

Given that from Eq. 13.4

$$\mu_{S^B_{\{\ \}}} = 0 \text{ and } \mu_{S^B_{\{u\}}} = \int_{-\infty}^{+\infty} xe\left(x, \frac{1}{\mu_{S^B_u}}\right) = \mu_{S^B_u}$$

and from Eq. 13.5

$$\sigma^2_{S^B_{\{\ \}}} = 0 \text{ and } \sigma^2_{S^B_{\{u\}}} = \int_{-\infty}^{+\infty} \left(x - \mu_{S^B_u}\right)^2 e\left(x, \frac{1}{\mu_{S^B_u}}\right) = \sigma^2_{S^B_u}$$

it follows from Eq. 13.6 that

$$S^I_a \approx N\left((1 - f_u)\mu_{S^B_u}; (1 - f_u)\sigma^2_{S^B_u}\right)$$

and from Eq. 13.7

$$L_{a_i} \approx N\left(\mu_{L^0_{a_i}} + (1 - f_u)\mu_{S^B_u}, \sigma^2_{L^0_{a_i}} + (1 - f_u)\sigma^2_{S^B_u} \right)$$

which is the same result as Eq. 13.2. This implies that indeed our approach for distribution systems summarized in Sect. 13.4 follows as a special case of the approach for assembly systems summarized in this section. This finding will help us in further expanding the approach to more generic network structures in the next section.

Extension to *n*-echelon Assembly Systems

In the next section, we'll formulate an algorithm for spanning tree networks. *N*-echelon assembly systems are a special case of a spanning tree network.

13.6 System Safety Stock Optimization in Generic Networks

13.6.1 Introduction of a Spanning Tree System

As introduced in Sect. 13.2 a spanning tree is a connected graph that contains *N* nodes and *N*–1 arc. Assembly networks and distribution networks are special cases of spanning trees. We make the following assumptions:

- A spanning tree is characterized by a matrix *U*, where u_{ij} is the number of units from the upstream product *i* per downstream unit *j*.
- We start labelling downstream in the supply chain to reflect customer focus and assume without loss of generality that $u_{ij} = 0$ for $i \le j$.
- Notice that if $u_{ij} \neq 0$, *i* is an immediate predecessor of *j*. We define the set of immediate predecessors of *j* as $\rho_j = \{i | u_{ij} > 0\}$.
- We define the set of starting nodes $S = \{i | u_{ij} = 0, \forall j\}$. These are the customer facing nodes, without successors. They show up as blank rows in the matrix *U*.
- We define the set of end nodes $\varepsilon = \{i | u_{ji} = 0, \forall j\}$. These are the most upstream nodes, without predecessors. They show up as blank columns in the matrix *U*.

Further, we assume:

- Independent demand occurs at the starting nodes only and is normally distributed, that is $d_i \sim N\left(\mu_{d_i}, \sigma^2_{d_i}\right), \forall i \in S$.

- Nominal lead times are normally distributed, that is $L_i^0 \sim N\left(\mu_{L_i^0}, \sigma_{L_i^0}^2\right)$. The nominal lead times assume all inputs are available to start an operation. The index 0 illustrates that assumption.
- An operation can require one or multiple inputs. As the preceding stages experience stock-outs, actual lead times differ from the nominal lead times. Actual lead times include an incoming service time which is the combined effect of the preceding stages experiencing stock-outs. Incoming service times are approximated by a normal distribution and denoted as $S_i^I \approx N\left(\mu_{S_i^I}, \sigma_{S_i^I}^2\right)$. The index *I* indicates that it is an incoming service time. The index *i* indicates it is to stage *i*. The actual lead times are also approximated by a normal distribution and denoted as $L_i \approx N\left(\mu_{L_i}, \sigma_{L_i}^2\right)$ (without the index 0). Actual lead times are to be used in the calculation of safety stocks and reorder points.
- For the starting stages $i \in S$:
  - given a normally distributed demand and a normal approximation for the actual lead-time, the demand during the actual lead-time can be approximated by a normal distribution, that is $D_i \approx N\left(\mu_{D_i}, \sigma_{D_i}^2\right)$, with $\mu_{D_i} = \mu_{d_i}\mu_{L_i}$ and $\sigma_{D_i}^2 = \mu_{L_i}\sigma_{d_i}^2 + \mu_{d_i}^2\sigma_{L_i}^2$,
  - we calculate safety stocks on the basis of a volume fill rate, that is $SS_i = k_i\sigma_{D_i}$ with k_i found from a target volume fill rate f_i through inversion of $f_i = 1 - \frac{1}{Q_i}\left[G\left(R_i, \mu_{D_i}, \sigma_{D_i}\right) - G\left(R_i + Q_i, \mu_{D_i}, \sigma_{D_i}\right)\right]$, with R_i the reorder point $R_i = \mu_{D_i} + SS_i$, Q_i a fixed lot size and the help function *G* defined as $G\left(r, \mu_D, \sigma_D\right) = \left(\mu_D - r\right)\left[1 - \Phi\left(r; \mu_D; \sigma_D^2\right)\right] + \sigma_D^2\varphi\left(r; \mu_D; \sigma_D^2\right)$,
  - the reorder process at the starting stages is based on a $\left(R_i, Q_i\right)$ policy with $R_i = \mu_{D_i} + SS_i$ the reorder point and Q_i a fixed lot size and
  - an order on upstream stages is to be delivered in full. In cases where multiple inputs are required, the stage can only start operation when all inputs are available.
- The reorder process at the starting stages allows characterizing dependent demand to their immediate predecessors:
  - in the case of n_i *identical* successors for a stage *i* (e.g. $n_i = 1$: linear or assembly, $n_i \neq 1$: n_i-identical retailer system), the dependent demand to an upstream stage can be characterized by a binomial distribution. The probability for success *p* is $\frac{\mu_{d_j}}{Q_j}$, with *j* an identical successor. The number of trials is n_i. In general the number of orders o_i for *i* is binomially

distributed with $\mu_{o_i} = \frac{\mu_{d_j}}{Q_j} n_i$ and $\sigma_{o_i} = \sqrt{n_i \frac{\mu_{d_j}}{Q_j}\left(1 - \frac{\mu_{d_j}}{Q_j}\right)}$. The demand distribution is derived in multiplying with the identical order size Q_i. We'll use the resulting average and standard deviation for a normal approximation of the demand for *I*, that is $d_i \approx N\left(\mu_{d_i}, \sigma_{d_i}^2\right)$ with $\mu_{d_i} = n_i \mu_{d_j}$ and $\sigma_{d_i} = Q_j \sqrt{n_i \frac{\mu_{d_j}}{Q_j}\left(1 - \frac{\mu_{d_j}}{Q_j}\right)}$. This is analogous to the demand for components in Sect. 13.4 and

 – in the case of non-identical successors for a stage *i*, we approximate demand by a normal distribution $d_i \approx N\left(\mu_{d_i}, \sigma_{d_i}^2\right)$ with $\mu_{d_i} = \sum_j u_{ij} \mu_{d_j}$ and $\sigma_{d_i}^2 = \sum_j u_{ij} \sigma_{d_j}^2$. In practical applications we would measure the average and variance of dependent demand.

- For upstream stages $i \notin S$:
  - given the normal approximation of the dependent demand and the actual lead-time, the demand during the actual lead-time can be approximated by a normal distribution and
  - safety stocks, reorder points, reorder policies, order allocation policies follow in analogy to the starting stages.
- For the end stages $i \in \varepsilon$:
  - we assume the outside suppliers to have an infinite stock. As such the nominal and actual replenishment lead-time are equal for the end stages and
  - demand during lead-time, safety stocks, reorder points, reorder policies, order allocation policies follow in analogy to the starting stages.
- We assume the following cost figures to be known:
  - C_i^{stage} is the cost added per stage;
  - this allows deriving the standard cost C_i^{std} in accumulating the stage cost of the predecessors stages and
  - $C_i^{holding}$ is the cost of holding a unit in stock for 1 year, typically calculated as a percentage of the standard cost C_i^{std}.

Example Spanning Tree System: Battery Supply Chain

In this section, we will work on an example system described by Graves and Willems (2003). It is modelled after a real-life supply chain for the manufacturing and distribution of batteries and is illustrated in Figure 13.19.

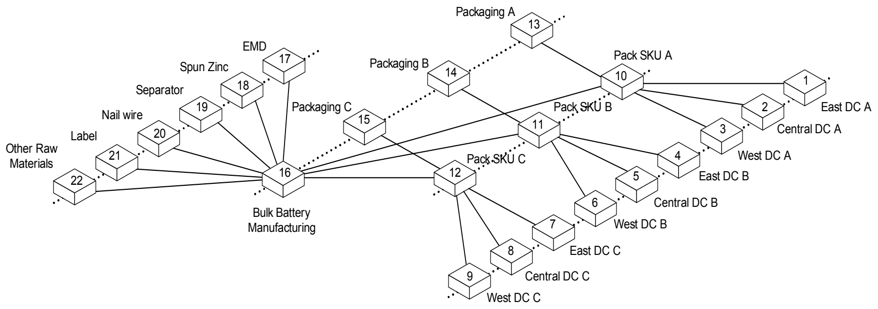


Fig. 13.19 The battery supply chain from Graves and Willems (2003)

A first echelon is the supply of raw materials, stages 17–22, and of packaging material, stages 13–15. This supply would typically be an ordering and delivery process managed by a purchasing department. Notice that only the important raw materials are modeled separately. Non-critical components, with ample supply, low cost and short lead times, are grouped into a category called "other raw materials". A second echelon is the production of batteries in bulk (stage 16). This is an assembly process, typically managed by a production department. The assembly operation can only start when all necessary components are available. A third echelon is the packaging of the bulk batteries into three variants A, B and C (stages 10–12). These are again assembly processes, typically managed by the same production department. A fourth echelon is the distribution of the packed SKUs A, B and C to three distribution centers: east, central and west (stages 1–9). These distribution processes are typically managed by a logistics department. Part of the distribution and warehousing may be outsourced. Planning and ownership of the inventory most often remain in the company. This example supply chain is quite rich in that it has purchasing, production and distribution characteristics or linear, converging and diverging subsystems. We believe that many such environments exist today and that they are not well integrated from a stock management perspective. For the example battery supply chain a possible matrix U becomes (see Table 13.3):

- verify that $u_{ij} = 0$ for $i \le j$ (above the diagonal);
- the set of end nodes can be derived as $\varepsilon = \{13,14,15,17,18,19,20,21,22\}$, columns with all zeros; and
- the set of starting nodes can be derived as $S = \{1,2,3,4,5,6,7,8,9\}$, rows with all zeros.

Table 13.3 The battery supply chain: example network matrix *U*

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Notice that for $u_{ij} > 0 \Rightarrow u_{ij} = 1;$ or units of measurement reflect the units of consumption. Graves and Willems (2003) provide real-life data for the nominal lead times $\mu_{L_i^0}$, added cost per stage C_i^{stage}, the resulting standard cost per stage C_i^{std}, average μ_{d_i} and standard deviation σ_{d_i} of the independent demand at the starting nodes. They are summarized in Table 13.4. Their assumption of the base-stock policy with common review cycle allows characterizing the dependent demand as $d_i(t) = \sum_j u_{ij} d_j(t)$ which for the average and variance means $\mu_{d_i} = \sum_j u_{ij} \mu_{d_j}$ and $\sigma_{d_i}^2 = \sum_j u_{ij} \sigma_{d_j}^2$. This allows filling out Table 13.4 to Table 13.5.

We make the following modifications and extensions. First we assume the coefficient of variation for lead times is 15%. Second, we assume a coefficient of variation for the independent demand of 25%. The values observed by Graves and Willems (2003) are higher than 100%. This makes the assumption of normality very weak. The assumption of normality of independent demand is exogenous to both models. As such, reducing the coefficient of variation to 25% focuses the comparison of both models on the endogenous assumptions. It also allows generating a demand pattern for use in a discrete-event simulation. Third, lot sizes are calculated using the EOQ formula. For purchasing and distribution stages (1–9, 13–15, 17–22) we assume an ordering cost of $30, for production stages (10–12,16) we use an ordering cost of $300. Further we assume $Q_i \geq Q_j$ if $u_{ij} \neq 0$. This increases the lot size for the packaging material stages 13, 14 and 15 and the raw material stages 17, 21 and 22. It avoids many smaller orders are placed by the stages 13–15, 17, 21 and 22 to meet the bigger lot sizes of the downstream stages 10–12 and 16. This reduces the order cost without impacting the inventory cost. We refer to Axsäter (2000) for a more detailed discussion of this rounding.

For all its shortcomings, the EOQ model does allow us to model the significantly higher setup costs we've encountered in many traditional production environments in comparison to the order costs in distribution environments. The base-stock policy of Graves and Willems (2000, 2003) and Ettl *et al.* (2000) does not allow that.

Table 13.4 The battery supply chain: data for demand, lead times and added cost per stage

Stage no.	Stage name	$\mu_{L_i^0}$	$\sigma_{L_i^0}$	C_i^{stage}	C_i^{std}	μ_{d_i}	σ_{d_i}
1	East DC A	4	0	0	0,82	67226	109308
2	Central DC A	6	0	0,02	0,84	43422	67236
3	West DC A	5	0	0,01	0,83	65638	119901
4	East DC B	4	0	0,01	0,96	15765	34079
5	Central DC B	6	0	0,01	0,96	16350	23277
6	West DC B	8	0	0,03	0,98	10597	23277
7	East DC C	4	0	0,01	1,20	6416	14125
8	Central DC C	4	0	0,01	1,20	5536	11213
9	West DC C	6	0	0,06	1,25	3519	6576
10	Pack SKU A	11	0	0,07	0,82		
11	Pack SKU B	11	0	0,12	0,95		
12	Pack SKU C	9	0	0,24	1,19		
13	Packaging A	28	0	0,16	0,16		
14	Packaging B	28	0	0,24	0,24		
15	Packaging C	28	0	0,36	0,36		
16	Bull battery manufacturing	5	0	0,07	0,59		
17	EMD	2	0	0,13	0,13		
18	Spun zinc	2	0	0,05	0,05		
19	Separator	2	0	0,02	0,02		
20	Nail wire	24	0	0,02	0,02		
21	Label	28	0	0,06	0,06		
22	Other raw materials	1	0	0,24	0,24		

Table 13.5 The battery supply chain: original data from Graves and Willems (2003)

Stage no.	Stage name	$\mu_{L_i^0}$	$\sigma_{L_i^0}$	C_i^{stage}	C_i^{std}	μ_{d_i}	σ_{d_i}
1	East DC A	4	0	0	0,82	67226	109308
2	Central DC A	6	0	0,02	0,84	43422	67236
3	West DC A	5	0	0,01	0,83	65638	119901
4	East DC B	4	0	0,01	0,96	15765	34079
5	Central DC B	6	0	0,01	0,96	16350	23277
6	West DC B	8	0	0,03	0,98	10597	23277
7	East DC C	4	0	0,01	1,20	6416	14125
8	Central DC C	4	0	0,01	1,20	5536	11213
9	West DC C	6	0	0,06	1,25	3519	6576
10	Pack SKU A	11	0	0,07	0,82	176286	175628
11	Pack SKU B	11	0	0,12	0,95	42712	57163
12	Pack SKU C	9	0	0,24	1,19	15471	19196
13	Packaging A	28	0	0,16	0,16	176286	175628
14	Packaging B	28	0	0,24	0,24	42712	57163
15	Packaging C	28	0	0,36	0,36	15471	19196
16	Bull battery manufacturing	5	0	0,07	0,59	234469	185691
17	EMD	2	0	0,13	0,13	234469	185691
18	Spun zinc	2	0	0,05	0,05	234469	185691
19	Separator	2	0	0,02	0,02	234469	185691
20	Nail wire	24	0	0,02	0,02	234469	185691
21	Label	28	0	0,06	0,06	234469	185691
22	Other raw materials	1	0	0,24	0,24	234469	185691

Table 13.6 The battery supply chain: data for the modified and extended problem

Stage no.	Stage name	$\mu_{L_i^0}$	$\sigma_{L_i^0}$	C_i^{stage}	C_i^{std}	μ_{d_i}	σ_{d_i}	Q
1	East DC A	4	0,6	0	0,82	67226	16807	71524
2	Central DC A	6	0,9	0,02	0,84	43422	10856	56795
3	West DC A	5	0,75	0,01	0,83	65638	16410	70248
4	East DC B	4	0,6	0,01	0,96	15765	3941	32011
5	Central DC B	6	0,9	0,01	0,96	16350	4088	32600
6	West DC B	8	1,2	0,03	0,98	10597	2649	25976
7	East DC C	4	0,6	0,01	1,20	6416	1604	18266
8	Central DC C	4	0,6	0,01	1,20	5536	1384	16967
9	West DC C	6	0,9	0,06	1,25	3519	880	13254
10	Pack SKU A	11	1,65	0,07	0,82	176286	25876	366264
11	Pack SKU B	11	1,65	0,12	0,95	42712	6266	167496
12	Pack SKU C	9	1,35	0,24	1,19	15471	2294	90070
13	Packaging A	28	4,2	0,16	0,16	176286	93850	366264
14	Packaging B	28	4,2	0,24	0,24	42712	31784	167496

Table 13.6 (continued)

Stage no.	Stage Name	$\mu_{L_i^0}$	$\sigma_{L_i^0}$	C_i^{stage}	C_i^{std}	μ_{d_i}	σ_{d_i}	Q
15	Packaging C	28	4,2	0,36	0,36	15471	14190	90070
16	Bull battery manufacturing	5	0,75	0,07	0,59	234469	26723	497977
17	EMD	2	0,3	0,13	0,13	234469	112219	497977
18	Spun zinc	2	0,3	0,05	0,05	234469	112219	540941
19	Separator	2	0,3	0,02	0,02	234469	112219	855303
20	Nail wire	24	3,6	0,02	0,02	234469	112219	855303
21	Label	28	4,2	0,06	0,06	234469	112219	497977
22	Other raw materials	1	0,15	0,24	0,24	234469	112219	497977

The lot sizes Q_i influence the characterization of the dependent demand. As introduced above, for stages with identical successors (13–15, 17–22) we apply the normal approximation $d_i \approx N\left(\mu_{d_i}, \sigma_{d_i}^2\right)$ with average and variance following from a binomial distribution as $\mu_{d_i} = n_i \mu_{d_j}$ and $\sigma_{d_i} = Q_j \sqrt{n_i \frac{\mu_{d_j}}{Q_j}\left(1 - \frac{\mu_{d_j}}{Q_j}\right)}$ with n_i, the number of identical successor stages, equal to one. For stages with non-identical successors (10–12, 16) we apply the normal approximation $d_i \approx N\left(\mu_{d_i}, \sigma_{d_i}^2\right)$ with average and variance following as $\mu_{d_i} = \sum_j u_{ij} \mu_{d_j}$ and $\sigma_{d_i}^2 = \sum_j u_{ij} \sigma_{d_j}^2$. The data for the modified and extended problem is shown in Table 13.6. $\mu_{L_i^0}$ and $\sigma_{L_i^0}$ characterize the nominal lead times, C_i^{stage} and C_i^{std} the cost structure, μ_{d_i} and σ_{d_i} the independent and dependent demand and Q_i the lot sizes. In what follows we will refer to Table 13.6 as the "modified and extended" problem *versus* the "original" problem of Graves and Willems (2003), summarized in Table 13.5. As in previous sections we will now first characterize the actual replenishment lead times and apply that to our example system.

13.6.2 Characterization of Actual Replenishment Lead Times in a Spanning Tree System

The algorithm below builds on the results of the previous subsection. To make the algorithm more comprehensive, we will illustrate it using the modified and extended version of the battery supply chain.

Iteration 0: Initialization

In the initialization we process all end stages $i \in \varepsilon$.

Step 1: Calculation of the Actual Lead Limes
For the end stages $i \in \varepsilon$ we assume:

- the incoming service times S_i^I are equal to zero, because of infinite supply; and
- as a result the actual lead times L_i equal the nominal lead times L_i^0.

Step 2: Calculation of the Back-order Service Times
Calculate the back-order service times for the end stages $i \in \varepsilon : S_i^B \approx E(\lambda_i)$ with $\lambda_i = \frac{1}{\mu_i}$ and μ_i following through numerical integration of Eq. 13.1.

Step 3: Adjust set of Processed Nodes
If we define a set of processed nodes after iteration *n* as P_n, then after initialization $P_0 = \varepsilon$.

Iteration n

It processes all stages *j* for which $\rho_j \subseteq P_{n-1}$. In the first iteration these are the stages for which all immediate predecessors are end nodes.

Step 1: Calculation of the Actual Lead Times
Calculate the actual lead-time for *j* using Eq. 13.4 where the incoming service time S_j^I can be calculated through Eq. 13.6 and where the back-order service time for a subset *A* follows from Eqs. 13.4 and 13.6. Notice that we are leveraging the findings of Sect. 13.4. We have shown there that the approach developed in Sect. 13.3 follows as a special case of our approach developed for assembly systems in Sect. 13.4 and can as such be applied to *j*, regardless of the number of stages in ρ_j.

Step 2: Calculation of the Back-order Service Times
Calculate the back-order service times for *j*, $S_j^B \approx E(\lambda_j)$ with $\lambda_j = \frac{1}{\mu_j}$ and μ_j following through numerical integration of Eq. 13.1.

Step 3: Adjust Set of Processed Nodes

$$P_n = P_{n-1} \cup \{ j | \rho_j \subseteq P_{n-1} \}.$$

Stop condition: Continue until P_n contains all stages.

We now apply the algorithm, step by step, to the modified and extended version of the battery supply chain.

Iteration 0: Initialization

The initialization processes the set of end nodes $\varepsilon = \{13,14,15,17,18,19,20,21,22\}$.

Step 0.1: Calculation of the Actual Lead Times
For the end stages we assume the incoming service time is zero. As a result the actual lead-time equals the nominal lead-time. This is shown in Table 13.7.

Table 13.7 Actual lead times battery supply chain: step 0.1

Stage no.	Stage name	$\mu_{L_i^0}$	$\sigma_{L_i^0}$	f_i	$\mu_{S_i^I}$	$\sigma_{S_i^I}$	μ_{L_i}	σ_{L_i}	μ_{d_i}	σ_{d_i}
1	East DC A	4	0,6							
2	Central DC A	6	0,9							
3	West DC A	5	0,75							
4	East DC B	4	0,6							
5	Central DC B	6	0,9							
6	West DC B	8	1,2							
7	East DC C	4	0,6							
8	Central DC C	4	0,6							
9	West DC C	6	0,9							
10	Pack SKU A	11	1,65							
11	Pack SKU B	11	1,65							
12	Pack SKU C	9	1,35							
13	Packaging A	28	4,2		0	0	28,00	4,20		
14	Packaging B	28	4,2		0	0	28,00	4,20		
15	Packaging C	28	4,2		0	0	28,00	4,20		
16	Bull battery manufacturing	5	0,75							
17	EMD	2	0,3		0	0	2,00	0,30		
18	Spun zinc	2	0,3		0	0	2,00	0,30		
19	Separator	2	0,3		0	0	2,00	0,30		
20	Nail wire	24	3,6		0	0	24,00	3,60		
21	Label	28	4,2		0	0	28,00	4,20		
22	Other raw materials	1	0,15		0	0	1,00	0,15		

Step 0.2: Calculation of the Back-order Service Times
Eq. 13.1 shows the back-order service time S_i^B depends on $(R_i, \mu_{d_i}, \sigma_{d_i}, \mu_{L_i}, \sigma_{L_i})$ with the reorder point R_i a function of $(f_i, Q_i, \mu_{d_i}, \sigma_{d_i}, \mu_{L_i}, \sigma_{L_i})$.

Table 13.8 shows $Q_i, \mu_{d_i}, \sigma_{d_i}$. We have not repeated them, to save some space in the tables. The actual lead times μ_{L_i}, σ_{L_i} have been calculated in step 0.1. For the service levels f_i, we could make any assumption at this moment. They are the decision variables in the safety stock optimization. We show the service levels (volume fill rates) that will lead to a zero safety stock for the upstream stages. For the stages with independent demand we will use a target service of 95%. Table 13.8 shows the back-order service times for the components and the packaging material (for zero safety stocks).

Table 13.8 Actual lead times battery supply chain: step 0.2

Stage no.	Stage name	$\mu_{L_i^0}$	$\sigma_{L_i^0}$	f_i	$\mu_{S_i^I}$	$\sigma_{S_i^I}$	μ_{L_i}	σ_{L_i}	μ_{d_i}	σ_{d_i}
1	East DC A	4	0,6							
2	Central DC A	6	0,9							
3	West DC A	5	0,75							
4	East DC B	4	0,6							
5	Central DC B	6	0,9							
6	West DC B	8	1,2							
7	East DC C	4	0,6							
8	Central DC C	4	0,6							
9	West DC C	6	0,9							
10	Pack SKU A	11	1,65							
11	Pack SKU B	11	1,65							
12	Pack SKU C	9	1,35							
13	Packaging A	28	4,2	58%	0	0	28,00	4,20	3,10	3,10
14	Packaging B	28	4,2	63%	0	0	28,00	4,20	3,44	3,44
15	Packaging C	28	4,2	67%	0	0	28,00	4,20	3,74	3,74
16	Bull battery manufacturing	5	0,75							
17	EMD	2	0,3	86%	0	0	2,00	0,30	0,37	0,37
18	Spun zinc	2	0,3	87%	0	0	2,00	0,30	0,37	0,37
19	Separator	2	0,3	92%	0	0	2,00	0,30	0,37	0,37
20	Nail wire	24	3,6	66%	0	0	24,00	3,60	2,64	2,64
21	Label	28	4,2	59%	0	0	28,00	4,20	3,02	3,02
22	Other raw materials	1	0,15	91%	0	0	1,00	0,15	0,22	0,22

Step 0.3: Adjust Set of Processed Nodes

$P_0 = \{13,14,15,17,18,19,20,21,22\}$ as illustrated in Figure 13.20.

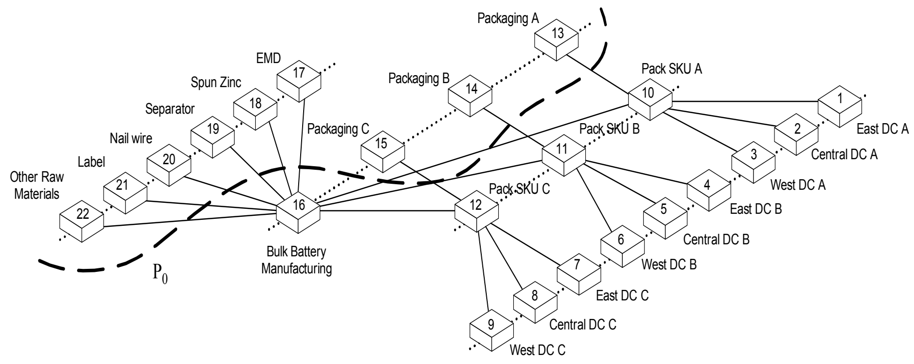


Fig. 13.20 Actual lead times battery supply chain: set of processed nodes after iteration 0

Iteration 1

It processes all stages j for which $\rho_j \subseteq P_0 = \{13,14,15,17,18,19,20,21,22\}$. From Figure 13.20 it follows this is a single stage, stage 16, the bulk battery assembly. For stages 10–12 $\rho_j \not\subset P_0 = \{13,14,15,17,18,19,20,21,22\}$ as stage $\{16\} \subseteq \rho_j$ and $\{16\} \not\subset P_0$.

Step 1.1: Calculation of the Actual Lead Times

We use Eq. 13.6 for calculating the incoming service time. S_{16}^I depends on the back-order service times of stages 17–22 and their service levels. The actual lead-time follows from the incoming service time and the nominal lead-time using Eq. 13.6. Table 13.9 shows the result. The inputs for the incoming service time are shown in bold.

Table 13.9 Actual lead times, battery supply chain: step 1.1

Stage no.	Stage name	$\mu_{L_i^0}$	$\sigma_{L_i^0}$	f_i	$\mu_{S_i^I}$	$\sigma_{S_i^I}$	μ_{L_i}	σ_{L_i}	μ_{d_i}	σ_{d_i}
1	East DC A	4	0,6							
2	Central DC A	6	0,9							
3	West DC A	5	0,75							
4	East DC B	4	0,6							
5	Central DC B	6	0,9							
6	West DC B	8	1,2							
7	East DC C	4	0,6							
8	Central DC C	4	0,6							
9	West DC C	6	0,9							
10	Pack SKU A	11	1,65							
11	Pack SKU B	11	1,65							
12	Pack SKU C	9	1,35							

Table 13.9 (continued)

Stage no.	Stage name	$\mu_{L_i^0}$	$\sigma_{L_i^0}$	f_i	$\mu_{S_i^I}$	$\sigma_{S_i^I}$	μ_{L_i}	σ_{L_i}	μ_{d_i}	σ_{d_i}
13	Packaging A	28	4,2	58%	0	0	28,00	4,20	3,10	3,10
14	Packaging B	28	4,2	63%	0	0	28,00	4,20	3,44	3,44
15	Packaging C	28	4,2	67%	0	0	28,00	4,20	3,74	3,74
16	Bull battery manufacturing	5	0,75		2,39	2,44	7,39	2,56		
17	EMD	2	0,3	**86%**	0	0	2,00	0,30	**0,37**	**0,37**
18	Spun zinc	2	0,3	**87%**	0	0	2,00	0,30	**0,37**	**0,37**
19	Separator	2	0,3	**92%**	0	0	2,00	0,30	**0,37**	**0,37**
20	Nail wire	24	3,6	**66%**	0	0	24,00	3,60	**2,64**	**2,64**
21	Label	28	4,2	**59%**	0	0	28,00	4,20	**3,02**	**3,02**
22	Other raw materials	1	0,15	**91%**	0	0	1,00	0,15	**0,22**	**0,22**

Step 1.2: Calculation of the Back-order Service Times

The calculation of the back-order service time for stage 16 follows in full analogy to that for the end stages. It depends on $\left(f_{16}, Q_{16}, \mu_{d_{16}}, \sigma_{d_{16}}, \mu_{L_{16}}, \sigma_{L_{16}}\right)$. For f_{16} we assume 66%, which will lead to a zero safety stock. Table 13.6 shows $Q_{16}, \mu_{d_{16}}, \sigma_{d_{16}}$. The actual lead-time $\mu_{L_{16}}, \sigma_{L_{16}}$ has been derived in step 1.1. The resulting back-order service time for stage 16 is shown in Table 13.10.

Table 13.10 Actual lead times, battery supply chain: step 1.2

Stage no.	Stage name	$\mu_{L_i^0}$	$\sigma_{L_i^0}$	f_i	$\mu_{S_i^I}$	$\sigma_{S_i^I}$	μ_{L_i}	σ_{L_i}	μ_{d_i}	σ_{d_i}
1	East DC A	4	0,6							
2	Central DC A	6	0,9							
3	West DC A	5	0,75							
4	East DC B	4	0,6							
5	Central DC B	6	0,9							
6	West DC B	8	1,2							
7	East DC C	4	0,6							
8	Central DC C	4	0,6							
9	West DC C	6	0,9							
10	Pack SKU A	11	1,65							
11	Pack SKU B	11	1,65							
12	Pack SKU C	9	1,35							
13	Packaging A	28	4,2	58%	0	0	28,00	4,20	3,10	3,10
14	Packaging B	28	4,2	63%	0	0	28,00	4,20	3,44	3,44
15	Packaging C	28	4,2	67%	0	0	28,00	4,20	3,74	3,74

Table 13.10 (continued)

Stage no.	Stage name	$\mu_{L_i^0}$	$\sigma_{L_i^0}$	f_i	$\mu_{S_i^I}$	$\sigma_{S_i^I}$	μ_{L_i}	σ_{L_i}	μ_{d_i}	σ_{d_i}
16	Bull battery manufacturing	5	0,75	66%	2,39	2,44	7,39	2,56	1,61	1,61
17	EMD	2	0,3	86%	0	0	2,00	0,30	0,37	0,37
18	Spun zinc	2	0,3	87%	0	0	2,00	0,30	0,37	0,37
19	Separator	2	0,3	92%	0	0	2,00	0,30	0,37	0,37
20	Nail wire	24	3,6	66%	0	0	24,00	3,60	2,64	2,64
21	Label	28	4,2	59%	0	0	28,00	4,20	3,02	3,02
22	Other raw materials	1	0,15	91%	0	0	1,00	0,15	0,22	0,22

Step 1.3: Adjust set of processed nodes
$P_1 = \{13,14,15,17,18,19,20,21,22\} \cup \{16\}$ as illustrated in Figure 13.21.

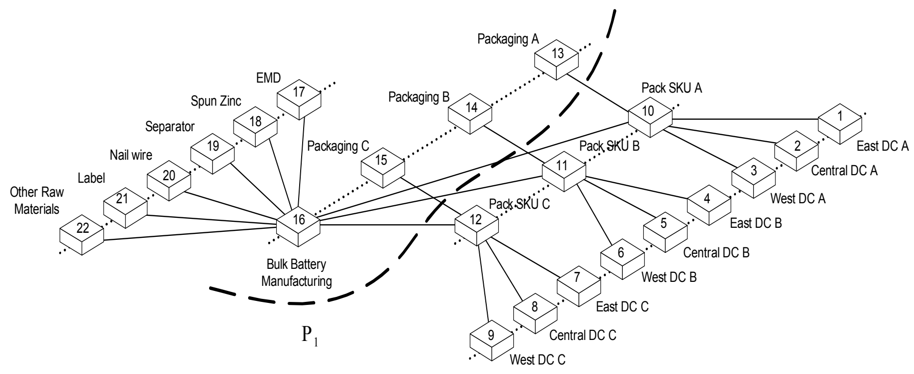


Fig. 13.21 Actual lead times battery, supply chain: set of processed nodes after iteration 1

Iteration 2

It processes all stages j for which $\rho_j \subseteq P_1 = \{13,14,15,17,18,19,20,21,22\} \cup \{16\}$. From Figure 13.21, it follows these are the stages 10, 11 and 12, the packaging of the bulk batteries in three types of packaging.

Step 2.1: Calculation of the Actual Lead Times

Using equation (Eq. 13.6), the incoming service times $S_{10}^I, S_{11}^I, S_{12}^I$ depend on the back-order service times and service levels of stages 13 and 16, 14 and 16, 15 and 16 respectively. The actual lead-time follows from the incoming service time and the nominal lead-time using equation (Eq. 13.7). Table 13.11 shows the results. Inputs for the incoming service time are again shown in bold.

Table 13.11 Actual lead times battery supply chain: step 2.1

Stage Nr	Stage Name	$\mu_{L_i^0}$	$\sigma_{L_i^0}$	f_i	$\mu_{S_i^I}$	$\sigma_{S_i^I}$	μ_{L_i}	σ_{L_i}	μ_{d_i}	σ_{d_i}
1	East DC A	4	0,6							
2	Central DC A	6	0,9							
3	West DC A	5	0,75							
4	East DC B	4	0,6							
5	Central DC B	6	0,9							
6	West DC B	8	1,2							
7	East DC C	4	0,6							
8	Central DC C	4	0,6							
9	West DC C	6	0,9							
10	Pack SKU A	11	1,65		1,85	2,21	12,85	2,76		
11	Pack SKU B	11	1,65		1,84	2,31	12,84	2,83		
12	Pack SKU C	9	1,35		1,82	2,73	10,82	2,73		
13	Packaging A	28	4,2	**58%**	0	0	28,00	4,20	**3,10**	**3,10**
14	Packaging B	28	4,2	**63%**	0	0	28,00	4,20	**3,44**	**3,44**
15	Packaging C	28	4,2	**67%**	0	0	28,00	4,20	**3,74**	**3,74**
16	Bull Battery Manufacturing	5	0,75	**66%**	2,39	2,44	7,39	2,56	**1,61**	**1,61**
17	EMD	2	0,3	86%	0	0	2,00	0,30	0,37	0,37
18	Spun Zinc	2	0,3	87%	0	0	2,00	0,30	0,37	0,37
19	Separator	2	0,3	92%	0	0	2,00	0,30	0,37	0,37
20	Nail Wire	24	3,6	66%	0	0	24,00	3,60	2,64	2,64
21	Label	28	4,2	59%	0	0	28,00	4,20	3,02	3,02
22	Other Raw Materials	1	0,15	91%	0	0	1,00	0,15	0,22	0,22

Step 2.2: Calculation of the Back-order Service Times
The calculation of the back-order service time for stages 10–12 follows in full analogy to that for the other stages. They depend on $\left(f_j, Q_j, \mu_{d_j}, \sigma_{d_j}, \mu_{L_j}, \sigma_{L_j}\right)$ with j = 10–12 respectively. For f_j, j = 10–12 , we could again make any assumption. For now we use 64%, 74% and 81% which will lead to zero safety stocks for stages 10-12 respectively. Table 13.6 shows $Q_j, \mu_{d_j}, \sigma_{d_j}$ for j = 10v12. The actual lead times μ_{L_j}, σ_{L_j} , j = 10–12, have been derived in step 2.1. The resulting back-order service times for stages 10–12 are shown in Table 13.12.

Step 2.3: Adjust Set of Processed Nodes
$P_2 = \{13,14,15,17,18,19,20,21,22\} \cup \{16\} \cup \{10,11,12\}$ as illustrated in Figure 13.22.

Table 13.12 Actual lead times battery supply chain: step 2.2

Stage no.	Stage name	$\mu_{L_i^0}$	$\sigma_{L_i^0}$	f_i	$\mu_{S_i^I}$	$\sigma_{S_i^I}$	μ_{L_i}	σ_{L_i}	μ_{d_i}	σ_{d_i}
1	East DC A	4	0,6							
2	Central DC A	6	0,9							
3	West DC A	5	0,75							
4	East DC B	4	0,6							
5	Central DC B	6	0,9							
6	West DC B	8	1,2							
7	East DC C	4	0,6							
8	Central DC C	4	0,6							
9	West DC C	6	0,9							
10	Pack SKU A	11	1,65	64%	1,85	2,21	12,85	2,76	1,76	1,76
11	Pack SKU B	11	1,65	74%	1,84	2,31	12,84	2,83	1,80	1,80
12	Pack SKU C	9	1,35	81%	1,82	2,73	10,82	2,73	1,73	1,73
13	Packaging A	28	4,2	58%	0	0	28,00	4,20	3,10	3,10
14	Packaging B	28	4,2	63%	0	0	28,00	4,20	3,44	3,44
15	Packaging C	28	4,2	67%	0	0	28,00	4,20	3,74	3,74
16	Bull battery manufacturing	5	0,75	66%	2,39	2,44	7,39	2,56	1,61	1,61
17	EMD	2	0,3	86%	0	0	2,00	0,30	0,37	0,37
18	Spun zinc	2	0,3	87%	0	0	2,00	0,30	0,37	0,37
19	Separator	2	0,3	92%	0	0	2,00	0,30	0,37	0,37
20	Nail wire	24	3,6	66%	0	0	24,00	3,60	2,64	2,64
21	Label	28	4,2	59%	0	0	28,00	4,20	3,02	3,02
22	Other raw materials	1	0,15	91%	0	0	1,00	0,15	0,22	0,22

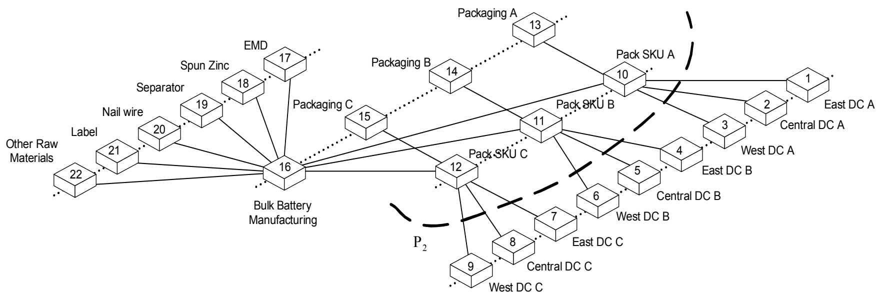


Fig. 13.22 Actual lead times, battery supply chain: set of processed nodes after iteration 2

Iteration 3

It processes all stages *j* for which
$\rho_j \subseteq P_2 = \{13,14,15,17,18,19,20,21,22\} \cup \{16\} \cup \{10,11,12\}$.

From Figure 13.22, it follows these are the stages 1–9.

Step 3.1: Calculation of the Actual Lead Times

Using Eq. 13.6 the incoming service times S_1^I - S_9^I depend on the back-order service time and service level of the stage 10, 11 or 12. The actual lead times follow from the incoming service time and the nominal lead-time using Eq. 13.7. Table 13.13 shows the results. Inputs for the incoming service time are again shown in bold.

Table 13.13 Actual lead times, battery supply chain: step 3.1

Stage no.	Stage name	$\mu_{L_i^0}$	$\sigma_{L_i^0}$	f_i	$\mu_{S_i^I}$	$\sigma_{S_i^I}$	μ_{L_i}	σ_{L_i}	μ_{d_i}	σ_{d_i}
1	East DC A	4	0,6		0,63	1,05	4,63	1,21		
2	Central DC A	6	0,9		0,63	1,05	6,63	1,39		
3	West DC A	5	0,75		0,63	1,05	5,63	1,29		
4	East DC B	4	0,6		0,48	0,93	4,48	1,10		
5	Central DC B	6	0,9		0,48	0,93	6,48	1,29		
6	West DC B	8	1,2		0,48	0,93	8,48	1,52		
7	East DC C	4	0,6		0,32	0,75	4,32	0,96		
8	Central DC C	4	0,6		0,32	0,75	4,32	0,96		
9	West DC C	6	0,9		0,32	0,75	6,32	1,17		
10	Pack SKU A	11	1,65	**64%**	1,85	2,21	12,85	2,76	**1,76**	**1,76**
11	Pack SKU B	11	1,65	**74%**	1,84	2,31	12,84	2,83	**1,80**	**1,80**
12	Pack SKU C	9	1,35	**81%**	1,82	2,73	10,82	2,73	**1,73**	**1,73**
13	Packaging A	28	4,2	58%	0	0	28,00	4,20	3,10	3,10
14	Packaging B	28	4,2	63%	0	0	28,00	4,20	3,44	3,44
15	Packaging C	28	4,2	67%	0	0	28,00	4,20	3,74	3,74
16	Bull battery manufacturing	5	0,75	66%	2,39	2,44	7,39	2,56	1,61	1,61
17	EMD	2	0,3	86%	0	0	2,00	0,30	0,37	0,37
18	Spun zinc	2	0,3	87%	0	0	2,00	0,30	0,37	0,37
19	Separator	2	0,3	92%	0	0	2,00	0,30	0,37	0,37
20	Nail wire	24	3,6	66%	0	0	24,00	3,60	2,64	2,64
21	Label	28	4,2	59%	0	0	28,00	4,20	3,02	3,02
22	Other raw materials	1	0,15	91%	0	0	1,00	0,15	0,22	0,22

Step 3.2: Calculation of the Back-order Service Times

The calculation of the back-order service time for stages 1–9 follows in full analogy to that for the other stages. They depend on $\left(f_j, Q_j, \mu_{d_j}, \sigma_{d_j}, \mu_{L_j}, \sigma_{L_j}\right)$ with j = 1–9 respectively. As these are stages facing independent demand, we assume 95% for f_j, j = 1–9. Table 13.6 shows $Q_j, \mu_{d_j}, \sigma_{d_j}$ for j = 1–9. The actual lead times

$\mu_{L_j}, \sigma_{L_j}, j$ = 1–9, have been derived in step 3.1. The resulting back-order service times for stages 1–9 are shown in Table 13.14.

Table 13.14 Actual lead times, battery supply chain: step 3.2

Stage no.	Stage name	$\mu_{L_i^0}$	$\sigma_{L_i^0}$	f_i	$\mu_{S_i^I}$	$\sigma_{S_i^I}$	μ_{L_i}	σ_{L_i}	μ_{d_i}	σ_{d_i}
1	East DC A	4	0,6	95%	0,63	1,05	4,63	1,21	0,56	0,56
2	Central DC A	6	0,9	95%	0,63	1,05	6,63	1,39	0,65	0,65
3	West DC A	5	0,75	95%	0,63	1,05	5,63	1,29	0,60	0,60
4	East DC B	4	0,6	95%	0,48	0,93	4,48	1,10	0,56	0,56
5	Central DC B	6	0,9	95%	0,48	0,93	6,48	1,29	0,64	0,64
6	West DC B	8	1,2	95%	0,48	0,93	8,48	1,52	0,75	0,75
7	East DC C	4	0,6	95%	0,32	0,75	4,32	0,96	0,53	0,53
8	Central DC C	4	0,6	95%	0,32	0,75	4,32	0,96	0,54	0,54
9	West DC C	6	0,9	95%	0,32	0,75	6,32	1,17	0,66	0,66
10	Pack SKU A	11	1,65	64%	1,85	2,21	12,85	2,76	1,76	1,76
11	Pack SKU B	11	1,65	74%	1,84	2,31	12,84	2,83	1,80	1,80
12	Pack SKU C	9	1,35	81%	1,82	2,73	10,82	2,73	1,73	1,73
13	Packaging A	28	4,2	58%	0	0	28,00	4,20	3,10	3,10
14	Packaging B	28	4,2	63%	0	0	28,00	4,20	3,44	3,44
15	Packaging C	28	4,2	67%	0	0	28,00	4,20	3,74	3,74
16	Bull battery manufacturing	5	0,75	66%	2,39	2,44	7,39	2,56	1,61	1,61
17	EMD	2	0,3	86%	0	0	2,00	0,30	0,37	0,37
18	Spun zinc	2	0,3	87%	0	0	2,00	0,30	0,37	0,37
19	Separator	2	0,3	92%	0	0	2,00	0,30	0,37	0,37
20	Nail wire	24	3,6	66%	0	0	24,00	3,60	2,64	2,64
21	Label	28	4,2	59%	0	0	28,00	4,20	3,02	3,02
22	Other raw materials	1	0,15	91%	0	0	1,00	0,15	0,22	0,22

Step 3.3: Adjust Set of Processed Nodes

$P_3 = \{13,14,15,17,18,19,20,21,22\} \cup \{16\} \cup \{10,11,12\} \cup \{1,2,3,4,5,6,7,8,9\}$

as illustrated in Figure 13.23.

Stop condition

It is clear that P_3 does now contain all stages. This stops the algorithm. All actual lead times have been calculated.

Table 13.15 shows the nominal lead times, the service level used in the calculations, the calculated actual lead times and then the service level and actual lead times as observed in a discrete-event simulation for the modified and extended battery supply chain.

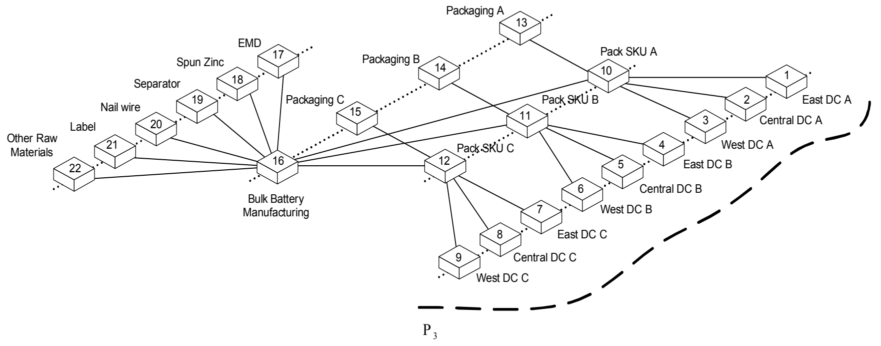


Fig. 13.23 Actual lead times, battery supply chain: set of processed nodes after iteration 3

Table 13.15 Actual lead times battery supply chain: calculation versus simulation results

Stage no.	Stage name	$\mu_{L_i^0}$	$\sigma_{L_i^0}$	f_i	μ_{L_i}	σ_{L_i}	$f_{i\,sim}$	$\mu_{L_i sim}$	$\sigma_{L_i sim}$
1	East DC A	4	0,6	95%	4,63	1,21	97%	4,52	1,22
2	Central DC A	6	0,9	95%	6,63	1,39	99%	6,38	1,22
3	West DC A	5	0,75	95%	5,63	1,29	98%	5,45	1,18
4	East DC B	4	0,6	95%	4,48	1,10	93%	4,20	1,08
5	Central DC B	6	0,9	95%	6,48	1,29	96%	6,31	1,20
6	West DC B	8	1,2	95%	8,48	1,52	96%	8,45	1,73
7	East DC C	4	0,6	95%	4,32	0,96	94%	4,39	1,24
8	Central DC C	4	0,6	95%	4,32	0,96	95%	4,25	1,05
9	West DC C	6	0,9	95%	6,32	1,17	97%	6,18	1,16
10	Pack SKU A	11	1,65	64%	12,85	2,76	76%	12,68	2,41
11	Pack SKU B	11	1,65	74%	12,84	2,83	79%	12,86	2,73
12	Pack SKU C	9	1,35	81%	10,82	2,73	88%	10,22	2,45
13	Packaging A	28	4,2	58%	28,00	4,20	37%	28,35	4,33
14	Packaging B	28	4,2	63%	28,00	4,20	46%	28,74	4,25
15	Packaging C	28	4,2	67%	28,00	4,20	63%	28,18	4,28
16	Bull battery manufacturing	5	0,75	66%	7,39	2,56	77%	7,05	1,42
17	EMD	2	0,3	86%	2,00	0,30	54%	1,97	0,30
18	Spun zinc	2	0,3	87%	2,00	0,30	53%	1,99	0,30
19	Separator	2	0,3	92%	2,00	0,30	73%	1,97	0,31
20	Nail wire	24	3,6	66%	24,00	3,60	63%	24,23	3,58
21	Label	28	4,2	59%	28,00	4,20	45%	28,25	4,72
22	Other raw materials	1	0,15	91%	1,00	0,15	0%	1,01	0,15

We analyze and discuss the results per group of stages.

- The end stages 10–12, 17–22:
  - actual lead times in the simulation show minor differences from those in the calculations: 1–2% for the average and 1–11% for the standard deviation. As for the end stages, by assumption, the actual lead times equal the nominal lead times, the differences of 1–2% and 1–11% are a measure for the error resulting from sampling lead-time distributions and
  - service levels in the simulation on average are lower than expected from the calculations. The probable cause is our normal approximations for demand deteriorate as we move up the network, leading to an underestimation of the demand variability.
- Stage 16:
  - the calculation (7,39; 2,56) provides a conservative estimate of the simulation results (7,05; 1,42). For the average that's a deviation of 5%, for the standard deviation of 80%; and
  - as we overestimate the actual lead-time, we overestimate the reorder point, which results in a higher service level in the simulation, 77%, than expected from the calculation, 66%.
- Stages 10–12 show comparable results:
  - calculations provide conservative but close estimates of the simulation results, 0–5% for the average and 4–12% for the standard deviation and
  - as we overestimate the actual lead-time, service levels in the simulation are higher than expected from the calculation.
- Stages 1–9:
  - on average the calculations provide a conservative estimate of the simulation results, with deviations of 0–2% for the average and 0-12% for the standard deviation. The deviations are in the order of the deviations shown to originate from sampling a lead-time distribution and
  - for stages 4 and 7 the simulated service levels of 93% and 94% are below the 95% target. Analysis of Table 13.6 shows that stage 7 has the biggest lot size of the stages 7–9 which compete for the inventory of stage 12. The full allocation policy at stage 12 will favor stages 9 and 8 respectively. This translates their service levels of 97% and 95% respectively. A comparable effect can be noted for stages 1–3 where the service levels from the simulation (97%, 99%, 98%) reflect the lot sizes of 71.524, 56.795 and 70.248 respectively, as shown in Table 13.6. For stage 4–6, Table 13.6 shows the lot sizes of stages 4 and 5 are very close. The fact that the service level of stage 4 is lower than that of stage 5 may be due to randomness in the simulation result.

In general, Table 13.15 indicates that the approximations are rather conservative and provide a close upper bound on the actual lead times as measured in the simulation.

13.6.3 System Safety Stock Optimization for a Spanning Tree System

As introduced above we calculate safety stocks as $SS_i = k_i \sigma_{D_i}$ with k_i found from a target volume fill rate f_i through inversion of $f_i = 1 - \frac{1}{Q_i}\left[G\left(R_i, \mu_{D_i}, \sigma_{D_i}\right) - G\left(R_i + Q_i, \mu_{D_i}, \sigma_{D_i}\right)\right]$, with R_i the reorder point $R_i = \mu_{D_i} + SS_i$, Q_i a fixed lot size, the help function G defined as $G(r, \mu_D, \sigma_D) = (\mu_D - r)\left[1 - \Phi\left(r; \mu_D; \sigma_D^2\right)\right] + \sigma_D^2 \varphi\left(r; \mu_D; \sigma_D^2\right)$, and with $\mu_{D_i} = \mu_{L_i}\mu_{d_i}$ and $\sigma_{D_i} = \sqrt{\mu_{L_i}\sigma_{d_i}^2 + \mu_{d_i}^2\sigma_{L_i}^2}$ characterizing the demand during lead-time. If we summarize SS_i is a function of $\left(f_i, Q_i, \mu_{d_i}, \sigma_{d_i}^2, \mu_{L_i}, \sigma_{L_i}^2\right)$ we can formulate the safety stock optimization problem as the minimization of:

$$\sum_i SS_i\left(f_i, Q_i, \mu_{d_i}, \sigma_{d_i}, \mu_{L_i}, \sigma_{L_i}\right) c_i^{holding} \tag{13.9}$$

μ_{L_i} and σ_{L_i} depend on $f_j, Q_j, \mu_{d_j}, \sigma_{d_j}, \mu_{L_j}, \sigma_{L_j}$ of all direct and indirect predecessor stages. The decision variables are the service levels f_j of the stages with dependent demand $j \notin S$. Changing a service level f_j impacts the actual lead times of all stages in the echelon of stage *j* and as such their safety stocks. The echelon of stage *j* consists of all direct or indirect successors of *j*. We minimize the inventory holding cost under the constraint of a given target service level f_i for the stages $i \in S$ facing independent demand. Table 13.16 shows the safety stock and the average inventory in pieces, in working capital (pieces times standard cost) and in inventory carrying cost (25% of the working capital). The average inventory is calculated as the safety stock plus half a lot size.

Table 13.16 System stock battery supply chain: calculation *versus* simulation results

Optimal solution	***SS* (pc)**	**Avg inv (pc)**	***SS* (wc)**	**Avg inv (wc)**	***SS* (icc)**	**Avg inv (icc)**
Finished product DC	397824	573738	343412	505145	85853	126286
Finished product plant		461736		410795		102699
Bulk battery		391740		231127		57782
Raw mat & packaging		3172575		300566		75141
Totals	**397824**	**4599790**	**343412**	**1447633**	**85853**	**361908**
Simulation results		**I_{sim} (pc)**		**I_{sim} (wc)**		**I_{sim} (icc)**
Finished product DC		531187		465573		116393
Finished product plant		387656		349299		87325
Bulk battery		319667		188603		47151
Raw mat & packaging		3061138		338372		84593
Totals		**4299648**		**134847**		**335462**

Table 13.16 (continued)

Single echelon base	***SS* (pc)**	**Avg inv (pc)**	***SS* (wc)**	**Avg inv (wc)**	***SS* (icc)**	**Avg inv (icc)**
Finished product DC	236487	410323	204823	364762	51206	91191
Finished product plant	439481	760797	373546	665045	93386	166261
Bulk battery	205108	459657	121014	271198	30253	67799
Raw mat & packaging	5030779	7304070	491178	703017	122795	175754
Totals	**5911855**	**8934847**	**1190561**	**2004022**	**297640**	**501006**

The "optimal solution" has been found through a grid search and corresponds with a zero safety stock for upstream stages. We compare it with the average inventory observed in the simulation and with a single-echelon base case assuming 95% service for all stages. Figure13.24 shows the resulting safety stock per echelon for the optimal and the single-echelon case.

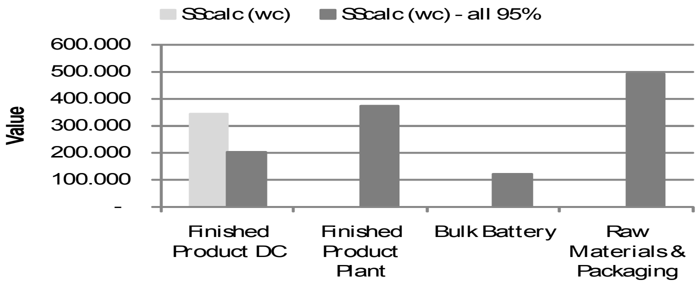


Fig. 13.24 System safety stock battery supply chain: calculation results

We believe it is not uncommon to find significant safety stocks upstream in the supply chain. Table 13.16 shows that from the single-echelon base case, the safety stocks can be reduced from 1,19 M$ to 0,34 M$. That is a staggering reduction of 71%. Figure 13.24 illustrates this is done while respecting the service target of 95% for the end customer. The required increase in the RDC safety stock of finished product is from 0,21 M$ to 0,34 M$ or a 68% increase. This may be hard to sell in an environment with decentralized control. Moreover, the effect on the average inventory is somewhat smaller due to the effect of the lot size. Still the average inventory can be reduced from 2.0 M$ to 1.4 M$ which is 28%.

13.6.4 Extension to Generic Systems

The algorithm introduced above can be extended beyond spanning tree networks. The basis of the algorithm is the extension of the "set of processed stages" P through iterations done for all stages j for which $\rho_j \subseteq P$, i.e. for which all immediate predecessor stages are in *P*. If we indicate the iterations by roman numbers, Figure 13.25 shows the iterations for the battery supply chain discussed above.

In Figure 13.5 we have introduced an example network that does not classify as a spanning tree. It has five stages and six arcs. However, it can be processed with the above algorithm. Its iterations are shown in Figure 13.26. The end nodes are the starting stages, processed in iteration 0. Iteration 1 processes the node that only consumes components from the end stages. Iteration 2 processes the starting stages.

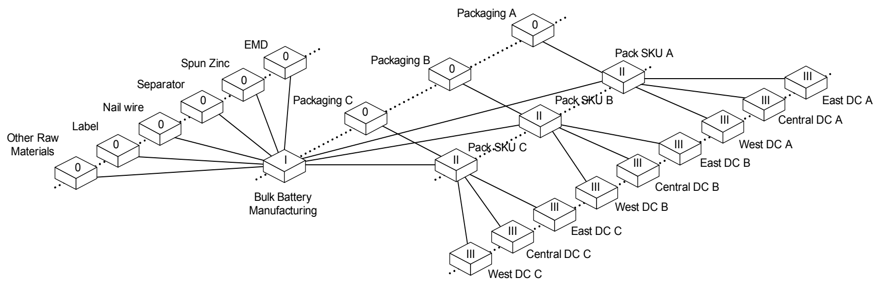


Fig. 13.25 Algorithm iterations for the battery supply chain

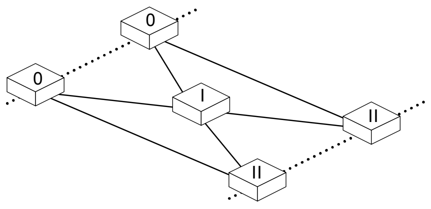


Fig. 13.26 Algorithm iterations for an example non-spanning tree

A more complex variant, like the one shown in the graph of Figure 13.27, can still be solved as shown by its algorithm iterations.

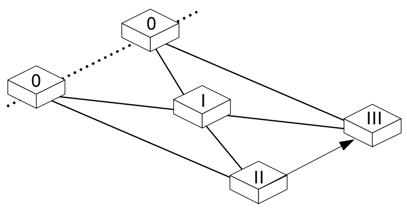


Fig. 13.27 Algorithm iterations for a second example non-spanning tree system

In practice, a wide range of production–distribution networks can be treated using the developed approach. An important topic left for future research is development of an appropriate optimization algorithm that can find the optimal solution. The bigger the network, the less efficient a grid search will be.

13.7 Conclusions

This chapter discusses some intelligent solution approaches used to optimize safety stocks in multi-echelon systems. Safety stock is a buffer against uncertainty stemming from demand and/or supply. The objective of a multi-echelon safety stock optimization problem is to minimize the system safety stock subject to a minimal service level for independent customer demand. Internal service levels for dependent demand are considered as variables. Reducing these internal service levels reduces the internal safety stock, but this requires some compensation at the downstream safety stock. In essence, multi-echelon safety stock models try to capture these dynamics.

Initially, safety stock optimization in distribution systems is investigated. An exponential approximation for the service time of back-orders in the central warehouse is primarily introduced. The order service time is defined as the time between order receipt and its allocation from available inventory. Next, a normal approximation for the actual replenishment lead-time of the retailers is introduced. The actual lead-time includes the nominal lead-time, which assumes that goods are available at the warehouse, and the service time for orders in the central warehouse. Finally, the approximation of the actual lead-time is used for the optimization of safety stock in distribution systems. These analytical results are also compared with the results of a discrete-event simulation.

Safety stock optimization in assembly systems is subsequently analyzed. From a safety stock point of view, an assembly system is more complex than a distribution system. Each stock point may have multiple predecessor stock points. The assembly can start only when all required inputs are available. Calculating the incoming service time to the assembly, which is defined as the time for all inputs to become available, requires analyzing all possible stock-out combinations and their resulting service time. A normal approximation for the incoming service time is introduced. The actual assembly lead-time combines the nominal lead-time, which assumes that all components are available, and the incoming service time. This is approximated using a normal distribution. This normal approximation of the actual assembly lead-time is used in the safety stock optimization of assembly systems. Notice that this approach for assembly systems can also be applied to distribution systems.

This approach for assembly systems is further extended to spanning tree systems. First an algorithm that calculates actual lead times in a spanning tree system is presented. For the calculation of service times the algorithm uses the results of assembly systems. In a next step the resulting actual lead times are used to formulate the safety stock optimization for a spanning tree system. Finally, the way the approach can be further extended from spanning tree systems to more generic systems is discussed. In all cases the analytical results were compared with simulation results to measure how good the proposed approximations are.

References

Axsäter S (1993) Continuous review policies for multi-level inventory systems with stochastic demand. In: Graves SC, Rinnooy Kan AHG, Zipkin PH (eds) Logistics of production and inventory, chapter 4, Elsevier Science Publishers, Amsterdam

Axsäter S (2000) Inventory control. Kluwer Academic Publishers, Boston

Buzby BR, Campbell GM, Webb I (1999) Cyclical schedules for one-warehouse, multi-retailer systems with dynamic demands. J. Oper. Res. Soc., 50:850–856

Campbell GM, Mabert V (1991) Cyclical schedules for capacitated lot sizing with dynamic demands. Mgmt. Sci., 37(4):409–427

Campbell GM (1996) Cyclic assembly schedules for dynamic demands. IIE Trans., 28(8):643–651

Chen F, Drezner Z, Ryan JK (2000) Quantifying the bullwhip effect in a simple supply chain: the impact of forecasting, lead times and information. Mgmt. Sci., 46(3):436–443

Clark AJ, Scarf H (1960) Optimal policies for a multi-echelon inventory problem. Mgmt. Sci., 6: 475–490

Crowston WB, Wagner MH, Williams JF (1973) Economic lot size determination in multi-stage assembly systems. Mgmt. Sci., 19(5):517–527

Crowston WB, Wagner MH (1973) Dynamic lot size models for multi-stage assembly systems. Mgmt. Sci., 20(1):14–21

de Kok AG (2000) Capacity allocation and outsourcing in a process industry. Int. J. Prod. Econ., 68:229–239

de Kok AG (1989) A moment-iteration method for approximating the waiting time characteristics of the GI/G/1 queue. Prob. Eng. Info. Sci., 3:273–287

de Kok TG, Visschers JWCH (1999) Analysis of assembly systems with service level constraints. Int. J. Prod. Econ., 59:313–326

de Kok TG, Fransoo JC (2003) Planning supply chain operations: definition and comparison of planning concepts. In: de Kok AG, Graves SC (eds) Supply chain management: design, coordination and operation, chapter 12, Elsevier Publishing Company

de Kok AG, Graves SC (2003) Supply chain management: design, coordination and operations. Elsevier Publisher

de Kok TG, Janssen F, van Doremalen J et al. (2005) Philips electronics synchronizes its supply chain to end the bullwhip effect. Interfaces, 35(1):37–48

Desmet B, Aghezzaf EH, Vanmaele H (2009a) A normal approximation model for safety stock optimization in a two-echelon distribution system. J. Oper. Res. Soc., January 7, doi: 10.1057/jors.2008.150

Desmet B, Aghezzaf EH, Vanmaele H (2009b) Safety stock optimization in two-echelon assembly systems: normal approximation models. Working paper, University of Ghent (submitted to Int. J. Prod. Res).

Deuermeyer B, Schwarz LB (1981) A model for the analysis of system service level in warehouse-retailer distribution systems: the identical retailer case. In: Schwartz LB (ed) Multi-level production/inventory control systems: theory and practice, 163–193, Elsevier Publishing Company

Diks EB, de Kok AG (1999) Computational results for the control of a divergent N-echelon inventory system. Int. J. Prod. Econ., 59(1–3):327–336

Ettl M, Feigin GE, Lin GY et al. (2000) A supply network model with base-stock control and service requirements. Oper. Res., 48(2):216–232

Federgruen A, Queyranne M, Zheng Y (1992) Simple power-of-two policies are close to optimal in a general class of production/distribution networks with general joint setup costs. Math. of Oper. Res., 17(4):951–963

Federgruen A, Zheng YS (1995) Efficient algorithms for finding optimal power-of-two policies for production/distribution systems with general joint setup costs. Oper. Res., 43(3):458–470

Federgruen A, Zheng YS (1993) Optimal power-of-two replenishment strategies in capacitated general production/distribution networks. Mgmt. Sci., 39(6):710–727

Graves SC (1988) Safety stocks in Manufacturing Systems. J. Manuf. Oper. Mgmt., 1: 67–101

Graves SC, Willems SP (1996) Strategic safety stock placement in supply chains. In: Proceedings of the 1996 MSOM conference, Dartmouth College, Hanover NH, June, pp.. 299–304

Graves SC, Willems SP (2000) Optimizing strategic safety stock placement in supply chains. Manuf. Ser. Oper. Mgmt., 2:68–83

Graves SC, Willems SP (2003) Supply chain design–safety stock placement and supply chain configuration. In: de Kok AG, Graves SC (eds) Supply Chain Management: Design, Coordination and Operation, chapter 3, Elsevier Science Publishers, Amsterdam

Groenevelt H, Johanssen J, Lederer P (1992) Cyclic planning. Available at:
http://www.ssb.rochester.edu/fac/lederer/SimonFacultyWeb/LedererPage.htm

Holland W, Sodhi MS (2004) Quantifying the effect of batch size and order errors on the bullwhip effect using simulation. Int. J. Logist.: Res. Appl., 7(3):251–261

Inderfurth K, Minner S (1998) Safety stocks in multi-stage inventory systems under different service measures. Eur. J. Oper. Res., 106:57–73

Kaplan U, Türkay M, Karasözen B et al. (2009) Optimization of supply chain systems with price elasticity of demand. J. Comp. doi: http://www3.iam.metu.edu.tr/bulent/makaleler/2009/ijoctemv.pdf

Köchel P, Nieländer U (2005) Simulation-based optimization of multi-echelon inventory systems. Int. J. Prod. Econ., 93–94 (8):505–513

Lee HL, Billington C (1993) Material management in decentralized supply chains. Oper. Res., 37:835–847

Lee HL, Padmanabhan V, Whang S (1997a) The bullwhip effect in supply chains. MIT Sloan Mgmt. Rev. Spring

Lee HL, Padmanabhan V, Whang S (1997b) Information distortion in a supply chain: the bullwhip effect. Mgmt. Sci., 43(4):546–558

Lin G, Ettl M, Buckley S et al. (2000) Extended-enterprise supply-chain management at IBM personal systems group and other divisions. Interfaces, 30 (1):7–25

McCullen P, Towill D (2002). Diagnosis and reduction of bullwhip in supply chains. Supp. Chain Mgmt. (An Int. J.), 7(3):164–179

Muckstadt JA, Roundy RO (1993) Analysis of multistage production systems. In: Graves SC, Rinnooy Kan AHG, Zipkin PH (eds) Logistics of production and inventory, chapter 2, Elsevier Science Publishers, Amsterdam

Muckstadt JA (2005) Analysis and algorithms for service parts supply chains. Springer publisher

Roundy R (1985) 98%-effective integer-ratio lot-sizing for one-warehouse multi-retailer systems. Mgmt. Sci., 31(11):1416–1430

Roundy R (1986) 98%-effective integer-ratio lot-sizing for a multi-product, multi-stage production/inventory system. Math. Oper. Res., 11(4):699–727

Roundy R (1989) Rounding off to powers of two in continuous relaxations of capacitated lot sizing problems. Mgmt. Sci., 35(12):1433–1442

Schwarz LB, Deuermeyer BL, Badinelli RD (1985) Fill-rate optimization in a one-warehouse n-identical retailer distribution system. Mgmt. Sci., 31(4):488–498

Sherbrooke CC (2004) Optimal inventory modeling of systems: multi-echelon techniques. Kluwer Academic Publishers

Slack N, Chambers S, Johnston R (2001) Operations management. Pearson Education

Swaminathan JM, Smith SF, Sadeh NM (1998) Modeling supply chain dynamics: a multi-agent approach. Dec. Sci., 29(3):607–632

Taylor DH (1999) Measurement and analysis of demand amplification across the supply chain. Int. J. Logist. Mgmt., 10(2):55–70

Tayur SR (1993) Computing the optimal policy for capacitated inventory models. Comm. Stoch. Models, 9:585–598

Van den Broecke F, Van Landeghem H, Aghezzaf E (2003) An application of cyclical master production scheduling in a multi-stage, multi-product environment. Prod. Plan. Control, 16(8):796–809

Van Houtum GJ (2006) Multi-echelon production/inventory systems: optimal policies, heuristics, algorithms. INFORMS Tutorial in Oper. Res. Series, 163–199

Chapter 14
Real-world Service Interaction with Enterprise Systems in Dynamic Manufacturing Environments

S. Karnouskos, D. Savio, P. Spiess, D. Guinard, V. Trifa and O. Baecker

Abstract The factory of the future will be heavily based on internet and web technologies. A new generation of devices with embedded hardware and software will feature greatly improved storage, computing, and networking capabilities. This will lead to a system landscape of millions of networked devices that is heterogeneous with respect to functionality but features standard interfaces. This new breed of devices will not only be able to store and report information about themselves and their physical surroundings, but execute more computations and local logic. They will form collaborative peer-to-peer networks and also connect to central systems. By eliminating media breaks, e.g. by replacing manual data entry with a direct connection to devices, this "internet of things" will feature end-to-end connectivity, making the models of the real world, as they exist in business systems, follow reality more precisely and with shorter delay. This will change the way we design, deploy and use services at all layers of the system, be it the device, line, plant, or company level or even between collaborating organizations. This chapter describes an architecture for effective integration of the services from the internet of things with enterprise services. We describe the case of centrally managing a population of devices that are located at different sites, including dynamic discovery of devices and the services they offer, near real-time cross-site interaction, interaction with business processes and distributed system management.

Keywords Service interaction, enterprise integration, dynamic manufacturing

S. Karnouskos (✉), D. Savio, P. Spiess, D. Guinard, V. Trifa and O. Baecker
SAP Research, SAP AG, Vincenz-Priessnitz-Strasse 1, D-76131 Karlsruhe, Germany
e-mail: stamatis.karnouskos@sap.com

14.1 Motivation

The last decade has witnessed a deep paradigm shift on the shop-floor where information and communication technologies are being used extensively. As high-performance micro-controllers are being embedded in devices used in manufacturing and process automation, services hosted on them will enable new applications that could significantly increase the efficiency and efficacy of current shop-floor systems.

As we are moving towards the "internet of things" as depicted by Fleisch and Mattern (2005), millions of interconnected devices will provide and consume information available on the network and cooperate. Service-oriented architecture (SOA) seems to be a promising solution to realize the necessary universal interoperability. SAP predicts that the world market for technologies, products, and applications that are related to the "internet of things" will increase significantly from €1.35 billion to more than €7.76 billion in 2012, with average annual growth rates of almost 50% (SAP, 2008).

In what we call "real-world SOA", each device:

- offers its functionality in a service-oriented way;
- is able to discover other devices and their hosted services dynamically at run-time;
- can invoke actions of the discovered services dynamically; and
- is able to publish and subscribe to typed, asynchronous events.

These distributed devices can be considered as a set of intelligent, proactive, fault-tolerant and reusable units (Colombo and Karnouskos, 2009) that can co-operate to form a dynamic infrastructure able to provide better insights of the current status of, e.g., a production line to the other higher levels in the factory information technology (IT) systems, such as manufacturing execution systems (MES). They also can dynamically react to business changes that can influence the production plan on the shop-floor.

As demonstrated in previous work (de Souza *et al.*, 2008; Jammes and Smit, 2005; Karnouskos *et al.*, 2007; Priyantha *et al.*, 2008), future shop-floor infrastructures can significantly benefit from service-oriented approaches, both in vertical (cross-level) and horizontal communication, as pursued within the SOCRADES project (www.socrades.eu). In these infrastructures, new, rich services can be created by orchestrating and combining services from different system levels, i.e. services provided by enterprise systems, by middleware systems on the network, and by devices themselves. The composed services with complex behaviour can be created at any layer (even at device layer). In parallel, dynamic discovery and peer-to-peer (P2P) communication allows for dynamically discovery of functionality of each device. The trend is to clearly move away from proprietary connections between monolithic hardware and software systems towards more autonomous systems that interact in a more standardized, cooperative and open way.

The convergence of applications and products towards the SOA paradigm improves shop-floor integration and transparency, thereby increasing reactivity and performance of the workflows and business processes commonly found in manufacturing and logistics. Events become available to any entity of the system as they happen, and business-level applications can exploit such timely information for purposes such as diagnostics, performance indications, or traceability. While these vertical collaborations are beneficial for business application software, new challenges arise: direct communication with devices can be error prone or unreliable, which must be considered when critical decisions, such as branches in a workflow, depend on it.

Business processes in a company are defined by the best practices of the respective industry and its goals. However, in reality, production processes are monolithic and system output is expected to be ideal. A production process instance (residing, e.g., in an MES system) usually has a series of vertical integrations with shop floor systems until it reaches the end of its lifetime. As a consequence, challenges arise when trying to make the processes adaptive or trying to extend it. Introducing process parameters to adapt to the dynamic nature of the shop-floor is tedious, especially for companies that span multiple production locations and heterogeneous IT systems.

The embedded device landscape is changing drastically as technology rapidly advances. Shop floors become populated with highly sophisticated networked embedded devices that have faster central processing units with lower energy consumption, yet are more compact. They can also do more than controlling local loops and can provide tools for real-time analysis (for example the CX1020 series programmable logic controller (PLC) of Beckhoff).

As devices can natively offer web services, they provide an interoperability layer that leads to easier coupling with other components despite of the high heterogeneity behind the web service facades. Device profile for web services (DPWS) Chan *et al.*, (2005), OPC-UA (Mahnke *et al.*, 2009) and representational state transfer (REST) are three of the emerging technologies for realizing web service-enabled devices. Thanks to the nature of web services, compositions of services can be easily created to match the desired scenario, very much like in state-of-the-art web mashups. Integration of devices at the functional level allows us to focus on orchestrating services based on their role in a process, and not the low-level interface of the device as such.

Service-based integration of shop-floor devices with enterprise systems brings many benefits in terms of business automation, response time, and data quality. Although these benefits make this integration highly desirable in a competitive economy, the unsupervised integration of devices with enterprise systems can also cause economic losses. These losses include: production halts, production time increase, reputation loss due to delays and even product recalls. When unexpected situations occur on the shop-floor, a rapid and dynamic adaptation of the business process is required in order to mitigate the effects that such an event can cause.

Hence, a beneficial integration of shop-floor devices with enterprise systems should provide characteristics that enable business processes to dynamically adapt to changes in the state of the device layer. With the current improvement of shop-floor devices and the adoption of SOA on all layers of the system, it is possible to create systems that are self-healing, self-monitoring, and self-optimizing.

Our aim is to depict how we can move towards highly dynamic manufacturing enterprises. Although simulations have been used before either at the shop-floor or at business process level, they have been used in an isolated way. In our work, we try to integrate the shop-floor and the business system and dynamically assess the state of the resulting, holistic system. With the help of simulations and monitoring, enterprises can adapt to situations and predict possible problems on the shop floor. SOA-based middleware, such as one we proposed in Karnouskos *et al.* (2007) and demonstrated in de Souza *et al.* (2008), has lead to preliminary promising results in this area. We propose to extend our SOA-based system to accommodate simulation and analytics as well as decision-making and process-mapping strategies. This helps enterprise systems to dynamically adapt to changes in the shop-floor, to reduce the gap between the real world and its digital representation, and also to optimize business processes.

14.2 Real-world Awareness

Significant effort has been invested into the integration of physical computing devices with standard enterprise software, such as enterprise resource planning (ERP) systems. Planning a production order or creating a bill of materials in the ERP application is neither effective nor optimized, unless the shop-floor is transparent. As an example, the manufacturing industry foresees enterprise applications to consider real-time events on the shop-floor to plan production, enhance customer relationship management, and have a healthy updated supply chain. This shop-floor intelligence obtained in real time allows business to adapt to the market demand and forecast shop-floor breakdowns in a timely fashion. Additionally as SOA approaches start to prevail (Kennedy *et al.*, 2008), the introduction cycle of new applications could be significantly shorter. This could enable exchange of real-time information across enterprises and trusted business partners, which will have an effect on the respective business decisions.

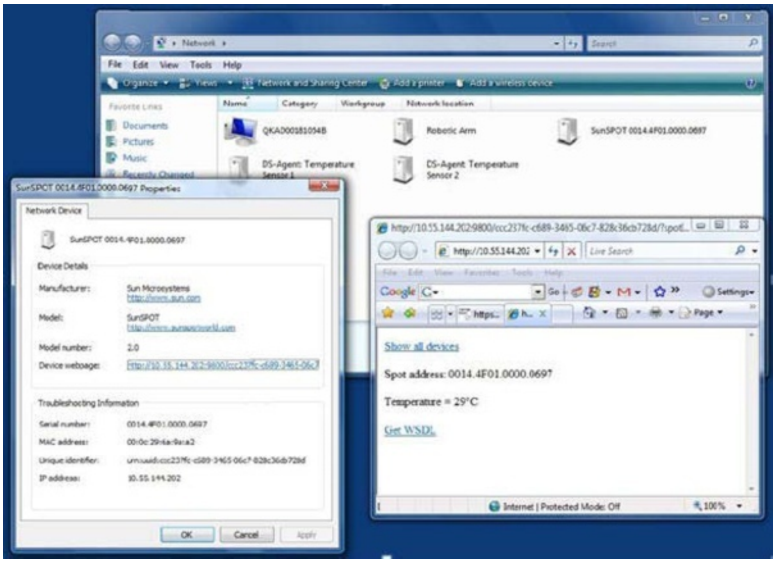
14.2.1 Device Integration Protocols

Shop-floor integration protocols based on web services as for example OPC UA (Mahnke *et al.*, 2009) are emerging in the automation domain. Until now, advanced features like dynamic device discovery, eventing and notification mechanisms are only on conceptual level in OPC UA specifications. Hence, OPC UA-based clients have to be installed on all systems that would need to consume shop-floor data from a device hosting the services. Furthermore the web service part of OPC UA is optional and performance evaluation or real-world experiences are not available to our knowledge.

Other shop-floor integration standards in the semiconductor industry are available, for example the SEMI (www.semi.org). Equipment Communications Standard/Generic Equipment Model (SECS/GEM) communication protocols connect a host computer and semiconductor manufacturing equipment. Photovoltaic Equipment Communication Interfaces is based on the SECS/GEM and targeted towards the photovoltaic industry. Clients have to be implemented that understand the protocols and communicate using interfaces defined in the standards. The biggest disadvantage is that only one client can interact with a server in a session. Multiple sessions are not possible. As such any approach to enable the service-oriented paradigm over this protocol would be very difficult to achieve without significant extensions.

Another standard dealing with ubiquitous device integration is DPWS (Chan *et al.*, 2005), which is a collection of web service standards. Initially, DPWS was conceived as a successor of universal plug-and-play (UpnP) for home automation scenarios, but recent work has shown its applicability to the automation world (Jammes and Smit, 2005). DPWS advances previous dynamic discovery concepts such as Jini (www.jini.org) and UPnP (www.upnp.org) to integrate devices into the networking world and make their functionality available in an interoperable way. DPWS is an effort to bring web services to embedded devices taking into consideration their constrained resources. Several implementations exist in Java and C (e.g. www.ws4d.org, www.soa4d.org), while Microsoft has also included a DPWS implementation (WSDAPI) by default in Windows Vista® and Windows Embedded CE operating systems. As depicted in Figure 14.1, any DPWS-empowered device can be dynamically discovered with existing Windows Vista® installations and its basic metadata can be read.

An alternative integration approach is REST (Fielding, 2000), which is the architectural principle that lies at the heart of the Web and shares a similar goal with integration techniques such as (DP)WS-* web services, that is increasing interoperability for a looser coupling between the parts of distributed applications. However, the goal of REST is to achieve this in a more lightweight and simpler manner; therefore it focuses on resources, and not functions as is the case with WS -* web services. In particular, REST uses the Web as an application platform and fully leverages all the features inherent to HTTP such as authentication, authorization, encryption, compression, and caching. This way, REST brings services "into the browser", i.e., resources can be linked and bookmarked and the results are visible with any web browser. There is no need to generate complex source code out of WSDL files to be able to interact with the service.


Network
Search
File Edit View Tools Help
Organize Views Network and Sharing Center Add a printer Add a wireless device
Favorite Links
Documents
Pictures
Music
Recently Changed
Name Category Workgroup Network location
QKAD00181054B
Robotic Arm
SunSPOT 0014.4F01.0000.0697
DS-Agent Temperature Sensor 1
DS-Agent Temperature Sensor 2
Live Search
Google
Settings
Page
Internet | Protected Mode: Off
100%
OK Cancel Apply

SunSPOT 0014.4F01.0000.0697 Properties

Network Device

SunSPOT 0014.4F01.0000.0697

Device Details

Manufacturer:	Sun Microsystems http://www.sun.com
Model:	SunSPOT http://www.sunspotworld.com
Model number:	2.0
Device webpage:	http://10.55.144.202:9800/ccc237fc-c689-3465-06c7

Troubleshooting Information

Serial number:	0014.4F01.0000.0697
MAC address:	00:0c:29:6a:9a:a2
Unique identifier:	urn:uuid:ccc237fc-c689-3465-06c7-828c36cb728d
IP address:	10.55.144.202

http://10.55.144.202:9800/ccc237fc-c689-3465-06c7-828c36cb728d/?spot...

Show all devices

Spot address: 0014.4F01.0000.0697

Temperature = 29°C

Get WSDL

Fig. 14.1 DPWS-based dynamic discovery of devices built in Windows Vista®

14.2.2 Device-to-Business Coupling

In the area of enterprise application integration, a few projects have explored the use of "mashups", also known as user-generated composite applications, to enable more flexible software composition within (and outside) the enterprise (Hoyer *et al.*, 2008; Liu *et al.*, 2007). However, they mainly focus on mashing up on-line services and do not address the issues and requirements that come with a physical world integration (e.g., as discussed in de Souza *et al.*, 2008; Marin-Perianu *et al.*, 2007). To ensure interoperability across all systems, recent work has focused on applying the concept of SOA; in particular web services standards (SOAP, WSDL, etc.) directly on devices (Jammes and Smit, 2005; Karnouskos *et al.*, 2007; Priyantha *et al.*, 2008). Implementing WS-* standards on devices presents several advantages in terms of end-to-end integration and programmability by reducing the need for gateways and mediators between the components. This enables the direct orchestration of services running on devices, with high-level enterprise services. As an example, if sensors physically attached to shipments could offer their functionality via web services; they could be easily discovered and integrated in a process that updates the status and location of the shipment directly in the involved ERP systems.

As an alternative to WS-* standards, more "web-oriented" (sometimes called "web of things" as opposed to the "internet of things" (Guinard and Trifa, 2009)) approaches are emerging, i.e. approaches (re)using standards on the web and applying them to real-world devices for them to provide services directly on the web. As an example, web feeds have been used to access data provided by sensor nodes (Dickerson *et al.*, 2008). In particular, they describe an extension to RSS better suited to accommodate high-rate data streams with a web-oriented querying interface to retrieve sensor data. A direct consequence of the stream abstraction is that sensors are considered solely as data publishers, not as service providers. Further examples (Drytkiewicz *et al.*, 2004; Guinard *et al.*, 2009; Luckenbach *et al.*, 2005; Wilde, 2007) propose and evaluate a RESTful (Fielding, 2000) architecture for sensor networks.

14.2.3 Integrating Heterogeneous Devices

In de Souza *et al.* (2008) and Karnouskos *et al.* (2007) we proposed an extensible integration architecture based on web services and capable also of supporting legacy products. There are basically three directions we follow as shown on Figure 14.2:

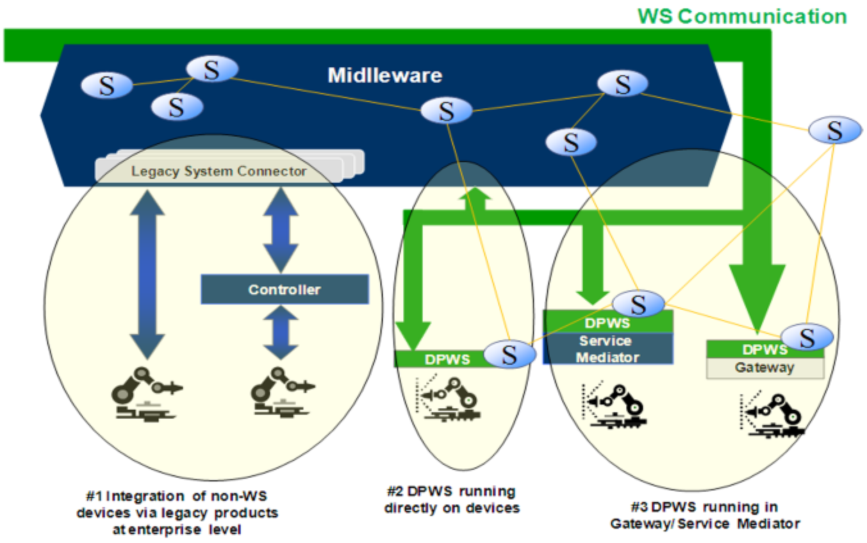

WS Communication
Midlleware
S
Legacy System Connector
Controller
DPWS
DPWS
Service
Mediator
DPWS
Gateway
#1 Integration of non-WS devices via legacy products at enterprise level
#2 DPWS running directly on devices
#3 DPWS running in Gateway/Service Mediator

Fig. 14.2 Integration approaches to couple legacy and emerging device infrastructure to enterprise systems

- Integration via legacy products: several tools are available to the market today, so we rely to them for providing the connectivity (as it is done today).
- Integration via gateways and service mediators: legacy or resource-scarce devices have their functionality wrapped as a web service at a higher layer, e.g., a control point (gateway approach) or their functionality is aggregated/composed

and a new service depending on (possibly many devices) is created (mediator approach).

- Web service enabled devices: these have the computational power and communication capabilities to natively run a stack that provides their functionality as a set of web services. These devices can be directly discovered and can interact with each other.

14.3 Enterprise Integration

The SOCRADES integration architecture (SIA as depicted in Figure 14.3) enables enterprise-level applications to interact with and consume data from a wide range of networked devices using a high-level, abstract interface that features web service standards. Those standards already constitute the standard communication method used by the components of enterprise-level applications. Web services are the technology canonically used to implement business processes, which are frequently modelled as orchestrations of available web services. This allows the connected, networked devices to directly participate in business processes while neither requiring the process modeller nor the process execution engine to know about the details of the underlying hardware.

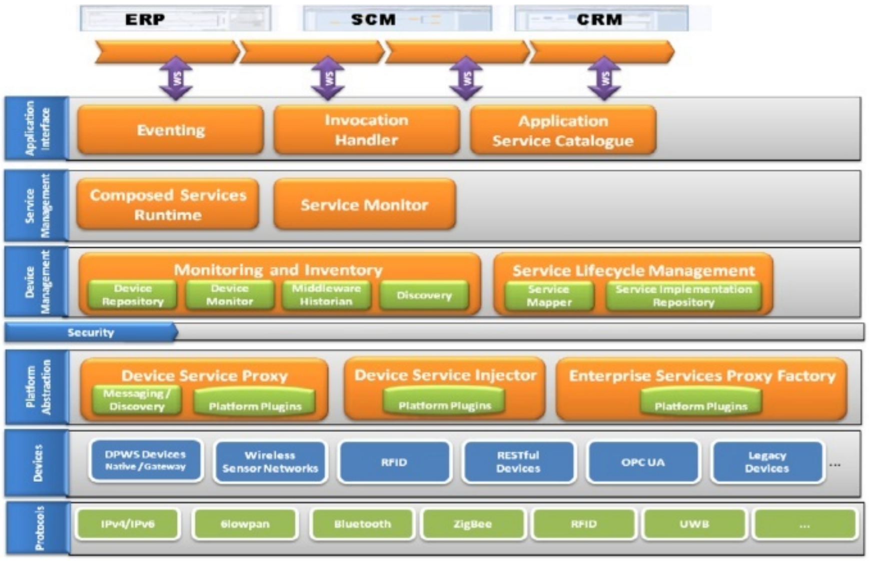


Fig. 14.3 The SOCRADES integration architecture

The requirements we want to cover as well as the functionality to be realized is analysed in de Souza *et al.* (2008) and Karnouskos *et al.* (2007). The architecture implemented hides the heterogeneity of hardware, software, data formats, and communication protocols that is present in today's embedded systems. The following layers can be distinguished: application interface, service management, device management, security, platform abstraction, and devices.

Application interface: this part of the integration architecture features a messaging (or eventing) system, allowing an application to consume any events whenever it is ready to and not when a low-level service happens to send them. A so-called invoker allows buffering invocations to devices that are only intermittently connected. Finally, a service catalogue enables human users and applications to find service descriptions and pointers to running service instances. Both atomic services hosted by the devices and higher-level composed services are listed here.

Service management: all functionality offered by networked devices is abstracted by services. Either devices offer services directly or their functionality is wrapped by a service representation. On this layer of the integration architecture and on all layers above, the notion of devices is abstracted from and the only visible assets are services. An important insight into the service landscape is to have a repository of all currently connected service instances. This is provided by the service monitor.

This layer also provides a run-time for the execution of composed services. We support the composition of business processes *primarily* by offering an execution service for underspecified BPEL processes, meaning that service compositions can be modelled as business processes where the involved partners do not need to be explicitly specified at design time.

Device management: all devices are dynamically discovered, monitored and their status is available to the enterprise services. Furthermore, it is possible to remotely deploy new services during run-time, in order to satisfy application needs.

Security: both devices and back-end services may only be accessed by clients that have a certain role and provide correct credentials that authenticate themselves. This layer implements the correct handling of security towards the devices and the enterprise-level applications.

Platform abstraction: as stated before, devices either offer services directly or their functionality is wrapped into a service representation. This wrapping is actually carried out on the platform abstraction layer. In the best case, a device offers discoverable web services on an internet protocol (IP) network. In this case, no wrapping is needed because services are available already. If the device type, however, does not have the notion of a service (it might use a message-based or data-centric communication mechanism), the abstraction into services that offer operations and emit events can be a complex task. In addition to service-enabling the communication with devices, this layer also provides a unified view on remotely installing or updating the software that runs on devices and enables the devices to communicate natively, i.e. in their own protocol with back-end devices.

Devices: heterogeneous devices are expected to connect to the architecture. These include industrial devices, home devices, or IT systems such as mobile phones, personal digital assistants, production machines, robots, building automation systems, cars, sensors and actuators, radio-frequency identification (RFID) readers, barcode scanners, or power meters. We used several of the listed types of devices during prototype implementations as shown in the demonstration section.

A single lightweight component called the local discovery unit (LDU) is implemented. It can be downloaded or deployed on any Java-supported device. This component dynamically discovers devices with respect to the protocols it supports and connects them to SIA. We can therefore realize very dynamic scenarios, where an LDU for IP is downloaded and run in a mobile phone, and via its WiFi interface dynamically finds and uses the functionality of all DPWS-based devices on the local network. Several LDUs can be deployed on a network allowing for load balancing as well as wide protocol coverage. Finally, using diverse LDUs, the SIA is able to realize cross-network and even cross-enterprise dynamic device discovery and act as mediator of their services. Apart from the LDU, the rest of the components have been implemented based on open web service standards in Java EE 5 on the SAP NetWeaver CE platform, using EJB 3.0 and JPA.

Protocols: We expect that heterogeneity in communication protocols will exist also in the future as it serves different domains and respectively devices.

14.4 Integrating Manufacturing Equipment with the SOCRADES Integration Architecture

Traditionally, shop-floor data has been integrated with enterprise applications like ERP using proprietary, custom made, individually developed technologies, e.g. in the form of file transfers and proprietary middleware layers.

Standard connectors from the OPC Foundation for MES and Distributed Control Systems lowered the complexity of integration by providing a unified data format, but each connection must still be statically tailored for a particular group of devices of a certain vendor because the standard does not define the meaning of data points. If the same resource has to be used from different system layers or in a P2P way by other systems on the shop floor, appropriate clients have to be developed for each interface. Additionally, the old OPC specifications were tightly coupled with the DCOM technology, making adopters dependent on the Windows family of operating systems from Microsoft. In 2006, the OPC Foundation started specifying a new interface standard, OPC UA, as a successor of the native OPC standards based on COM/DCOM and mitigating some of the mentioned pain points. However, the idea of using plain web service on devices, creating SOA on the shop floor, is not implemented well in the new version of the interface standard. The web services are specified as an optional method to access OPC UA interfaces. A binary transport mechanism is mandatory for OPC UA devices to communicate. It specifies a very complex metamodel for data and service modeling that makes the implementation of servers and clients quite challenging. The added value over plain web services is questionable. Well-documented web service interfaces could give the same value at a fraction of the complexity and still allow for cross-platform interoperability, flexibility, and dynamic processes.

In the business software domain, web services were originally designed to allow cooperation between applications within and across companies, possibly from different vendors, implemented using different programming languages, and running on different operating systems. They bring more flexibility and interoperability to cross-enterprise transactions that required constant change.

SAP ERP uses iDOCs to pass data between systems. iDOCs are (potentially large) extensible markup language (XML) documents, which contain data relevant for business transactions. In more recent versions of the application, using an enterprise SOA approach, these transactions are exposed as web services. A central universal description discovery and integration registry keeps track of services that can be consumed across the company. This enables ERP transactions to be more atomic, stand alone, and stateless. SAP ERP systems also provide services, which are harmonized with enterprise models based on process components, business objects and global data types. They follow well-defined common rules of business content. Web service-enabled devices on the shop floor would host services allowing them to directly monitor and influence physical processes. But the fragile nature of a device and its environment can influence the services hosted on the device. They lead to differences in reliability and availability (compared to regular enterprise services), which have to be considered when integrating devices on the shop floor with ERP platforms:

- Devices can be mobile and feature wireless communication, so the services they offer can appear and disappear on the network (as they connect and disconnect).
- Mobile devices can be battery powered and can be unavailable between the time the battery is exhausted and the time it is replaced.
- Devices might only be able to consume a fraction of the data related to a business transaction. Many elements of the XML would be irrelevant for a device.
- *Vice versa*, a device (e.g. a temperature sensor) cannot be expected to deliver the correct business context (e.g. the ID of its location in an asset management application) for the data it reports (this can be expected in an ERP system).

Although deploying web services on devices better embeds them in a company's SOA, shop-floor services are more fine-grained and often concentrate on technical issues. To bridge between the data types and semantics used at device level and those used at ERP system level, it will in most cases be necessary to use MES or SOA-based middleware acting as gateways.

As different manufacturers offer devices on the shop floor, they cannot be expected to have the same micro-controllers, have their communication application programming interfaces (API's) implemented in the same language, or run on the same operating system. However, a production process often requires devices from different manufacturers to work together. To make this possible today, shop-floor integrators have to invest heavily in device drivers and gateway mechanisms that take time to develop, are expensive and are difficult to maintain. In the near future, if devices would use web services, a workflow process can be easily orchestrated using modelling languages; they can be easily changed, extended, and adapted to changing needs of the business. Following an SOA-based approach, machine functionality offered as services can be used as process steps and therefore easily orchestrated. Furthermore, different plant-level systems can gain access to data from the shop-floor.

Modern manufacturing networks often span across continents. Typical examples are semiconductor and automobile production. Both industries require a large number of production steps where some are executed by specialized companies, e.g., silicon wafers are etched and oxidized at one place whereas cutting and packaging is done at a different location. Sales and distribution are done by yet another partner while the design and IP on the product is often owned by a company located at another side of the globe. Bringing constant change on these highly optimized networks to adapt to market trends or to react to competition is a challenging task. It is the same difficulty faced when detailed monitoring has to be done on distributed, multi-party processes and resources, where the monitoring does not rely on manual data entry, but on actual, near real-time production and progress figures as they are reported by embedded software in the low-level devices of each partner. Existing supply chain platforms can hardly fulfil these requirements.

SOA on the shop floor offers such fine-grained, inter-organizational insight into production. Using standard modelling languages like BPEL or by modeling executable BPMN 2.0 processes in NetWeaver Composition Environment, workflows can be composed without any need for device drivers or knowledge about the underlying hardware. These workflows can be a part of a business process that interacts with business software or humans. Manufacturing consortia can now use shared shop-floor data, derive key performance indicators (KPIs) from business-to-business partners, and react to market trends faster than before. By analysing the securely shared shop-floor data and KPIs, the end-to-end process can now be globally optimized to achieve a better performance compared to local optimization at each participating entity.

The seamless integration of the shop-floor data into the collaborative production process also allows for flexible ways of handling critical events that can occur at the shop-floor at run-time, such as a machine breakdown. Such events can be made visible across the whole manufacturing chain among the business partners. Customers at the end of the chain can be informed about problems, e.g. delayed delivery or the cancellation of an order in a timely manner. This can reduce costs and enhance the long-term business relationships.

However, the example of error handling also reveals a strong requirement for successful adoption of the flexible production networks. For the participating organizations to be willing to share data that potentially reveals the structure and performance of their core operations, organizations either need to trust each other or one of them must have the power to dominate and dictate the conditions of collaboration. In both cases, secure and reliable communication between the entities is required.

14.5 Towards Dynamic Adaptation

The business world is highly competitive and to successfully tackle everyday's challenges, operational managers and executives demand high-dependability and wide visibility into the status of their business process networks. The latest is done usually via business KPIs. However, to react in a flexible and optimal way to changing conditions, real-time information must flow via all layers from the shop floor up to the business process level.

As shown in Figure 14.4, three different phases help us achieve the required level of flexibility:

- Sense: real-time monitoring over an event-based infrastructure for networked devices.
- Think: an autonomous decision support system (DSS) guarantees business continuity.
- Act: decisions taken from the DSS are enforced in the infrastructure.

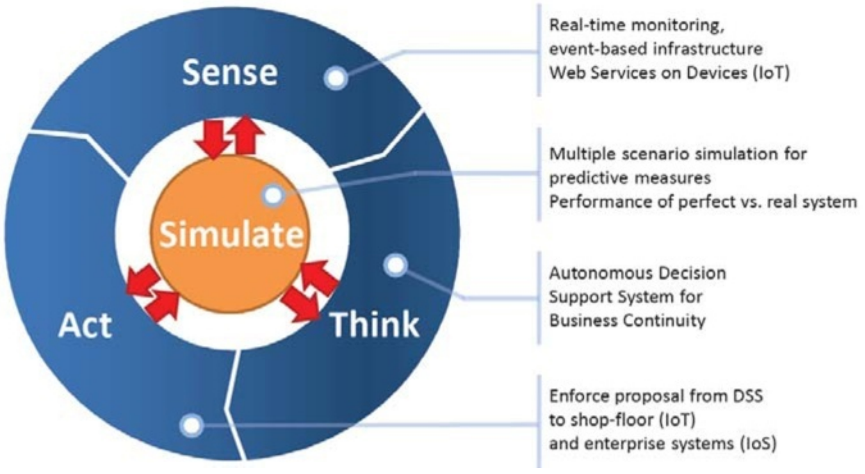

Sense
Simulate
Act
Think
Real-time monitoring, event-based infrastructure Web Services on Devices (IoT)
Multiple scenario simulation for predictive measures Performance of perfect vs. real system
Autonomous Decision Support System for Business Continuity
Enforce proposal from DSS to shop-floor (IoT) and enterprise systems (IoS)

Fig. 14.4 Towards autonomous systems via cross-layer monitoring, simulation and management

In the centre of the approach is the simulation that provides predicted system behaviour that is continuously compared with real-world results. The key technical concept to the whole approach is a cross-layer communication in order to provide effective monitoring, simulation and management. In this approach we target two characteristics of an autonomous system:

- Self-healing: automatic discovery and correction of faults or possible preventive actions. The system can recover from well-known problems including those that can be dynamically identified based on the correlation of events (complex event processing).
- Self-optimization: automatic monitoring and control of resources of the system can be done, in order for the different components that recognize themselves in a goal-driven manner with respect to the environmental context they act on. In that case, early indicators can be correlated and emerging problems are easier to pinpoint.

Figure 14.5 depicts the architectural approach proposed in this chapter. The core idea is the dynamic connection of enterprise systems with the shop-floor assisted by a DSS.

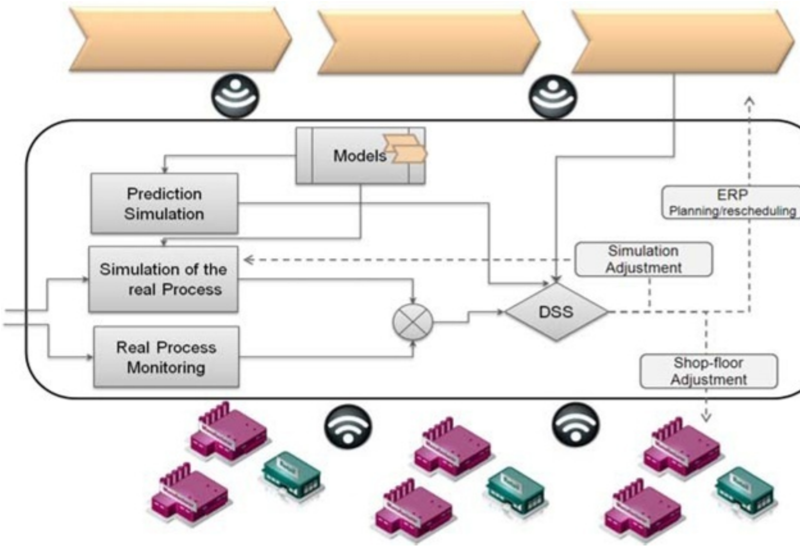


Fig. 14.5 Overview of the reactive system

The DSS takes into consideration dynamic data coming from monitoring the shop floor, running simulations and its results are given as input to the business process control, the shop-floor and even on fine-tuning the simulation itself. In this concept, continuous real-time data flow into a monitoring system. In parallel, a model of the shop-floor executes in a simulator. At specific intervals depending on the time or tasks, the output of the simulation and the monitoring are evaluated. Any deviation σ is used as input from the DSS. Parallel to σ, the DSS considers inputs from the enterprise systems as well as the prediction simulation, which predicts the next system state(s) of according to the existing models the system will continue to perform in the same way. The DSS considers all the input and makes decisions, e.g., for optimizing the performance of the system, preventing faults that would happen if the mode of operation is unchanged, etc. The DSS decisions are fed as input to the business systems, the shop-floor, and the simulation itself so that their behaviour can be adapted. Using precise information on the problems occurring (or predicted to occur) on the production line, the DSS can define measures to automatically modify the business process to heal the system. The result is that we are moving towards a self-* system that monitors and adapts itself according to the evaluation of the input from the sources mentioned.

14.5.1 Simulation

Simulating a process workflow as a production model on a computer takes away the risks of heavy investment. Modelling tools such as flowcharts, process mapping, and spreadsheets are often used to identify how the shop floor would look like for a particular business objective. However, such tools only show the relationships between processes and do not generally provide any quantitative performance measures. They are static and deterministic and do not consider the dynamics of real-life work in progress.

Dynamic production model simulation tools like WITNESS (Ahmed *et al.*, 2008) and ProModel (www.promodel.com) consider the dynamic characteristics of production, e.g. process flow, processing times, setup requirement, labour, control rules, breakdown, shift, loading schedule, etc. We propose to go one step further by using the simulation results and comparing them to the live input coming from the shop floor in order to assess the situation, proactively determine problem zones, and optimize the shop-floor.

Figure 14.6 presents the generic states of a machine in a production line. Based on a known machine state, a deterministic action can be performed. Nevertheless, defining the current state of the machine can be a challenging task. The information available to the back-end system consists of a great amount of events triggered through the assembly process. They could be identical to the simulated set of events and states, or deviate from the simulated conditions. Therefore, it is necessary to process these events in order to identify patterns that indicate which is the current state of the production line.

The advantage of production model simulation tools is that by changing the characteristics of production, the results on the shop floor can be accurately determined (Pidd and Robinson, 2007). These tools give a good in-depth understanding of how the manufacturing process would react to different situations on the shop floor. Such results can be considered as a reference for a series of deterministic characteristics of the shop floor (Ahmed *et al.*, 2008). As depicted in Figure 14.5, we propose a methodology where the workflow of a particular production process is continually monitored and compared to (pre)simulated results.

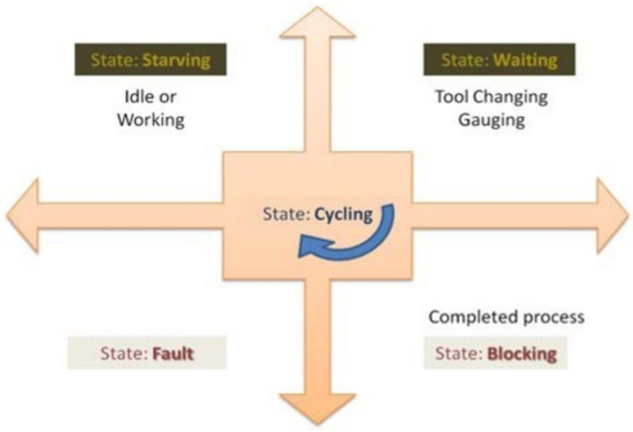


Fig. 14. 6 Machine states

The state of the actual workflow process is continually monitored and compared with the simulated result on the shop floor. Based on the possible deviation σ, the workflow process or the business process can be affected. The deviations of the shop floor behaviour from the expected result of the simulation have to be categorized in order to determine if the current condition would be tolerable or it would lead to a critical bottleneck. Various algorithms and methodologies can be combined to identify and categorize the state of the current production line.

14.5.2 Self-healing Mechanisms

Through a comparison between the expected state (simulation results) and the real state of the shop floor, it is possible to identify malfunctions in the system. This information is essential for the system to self-heal.

As an example, consider a knitting factory where T-shirts are manufactured through sequential steps like cutting, assembly, buttoning, quality check, and packaging. For instance, as shown in Figure 14.7, if the machine responsible for executing the process "Cutting 1" fails, the production order of the T-shirts with quality 3 will stop. This can delay the order delivery, cause reputation loss, and even breach a contract, which implies high costs to the factory.

If the manufacturer defines a maximum cost for the T-shirt production, it is possible to re-map the business process to avoid a production halt considering the new state of the system. This process re-mapping is depicted in Figure 14.7. With the modification in the business process, the system self-heals and continues the production, while a maintenance workflow is triggered. Although the item cost increases, it remains within the threshold specified by the manufacturer, and prevents major losses due to production halts.

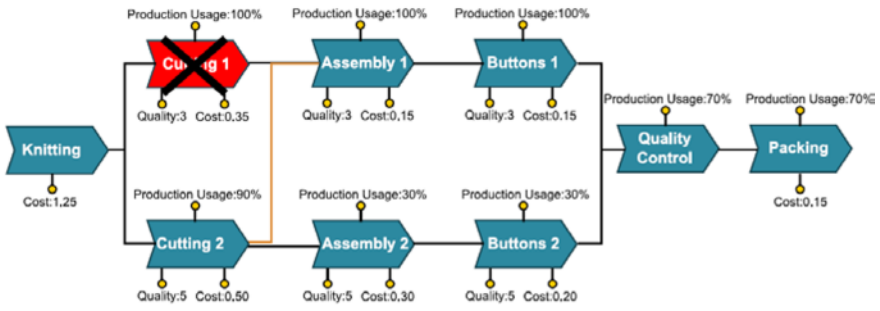


Fig. 14. 7 Self-healing and self-optimizing process re-route

Another self-healing mechanism investigated by this project explores predictive maintenance and possible production bottlenecks. Based on real-time data from the shop-floor and having identified the current status of the production line, it is possible to predict the course of the current production. This is done based on analysis of the previous production history and also based on the simulated production model. The result of such prediction model analysis is forwarded to a DSS, which then reacts by providing input to the business process modeller (Figure 14.5). Finally, the integration of ERP business processes and such DSSs can be performed through the SAP Manufacturing Intelligence and Integration (SAP MII) tool.

14.5.3 Self-optimizing Mechanisms

Business processes are available as services in the enterprise service repository. Hence, a set of rules can be modelled in the DSS to invoke a corresponding business process at the prediction of a critical bottleneck state on the shop floor. Alternatively, shop-floor devices hosting web services can be effectively used in such scenarios to prevent malfunction or breakdown of the machine. The DSS can reduce the production cycle on a particular assembly line when the system foresees a non-linear increase in temperature or a production variable on the workflow.

We propose to base the optimization process on swarm-intelligent (SI) principles (Bonabeau *et al.*, 1999). These methods were originally inspired by observation of various natural phenomena, in particular the collective behaviour of social insects and flocking and schooling in vertebrates. The application of SI to distributed, real-time, embedded systems aims at developing robust task-solving methodologies by minimizing the complexity (including the intelligence) of the individual units (in our case machines of the assembly line) and emphasizing parallelism, and self-organization. From an engineering standpoint, the principal advantages of SI system design are four-fold: scalability, from a few to thousands of units; flexibility, as units can be dynamically added or removed without explicit reorganization; robustness, not only through unit redundancy but also through an adequate balance between explorative and exploitative behaviour of the system, and simplicity (and low-cost) at the individual level, which also increases robustness. These properties would be highly beneficial if applied to machine production lines, and could be further optimized when machine have access to global information about the whole manufacturing process.

In particular, we propose to use threshold-based algorithms (TBA) for a flexible task allocation mechanism to decide of the dynamic path in the production line. TBA have been initially used to model the dynamic task allocation decision process in ant colonies, and has been successfully applied for example for power-aware optimized load balancing (Cianci *et al.*, 2005). Using TBA at the production-line level enables a reactive and fully decentralized decision process done dynamically by the machines at run-time, based both on the objects to process, external data (environmental data, priority of the tasks, market values, etc.), and the proprioceptive data of the machines. TBA model group behaviour based on a small number of control parameters (thresholds) that affect whether or not a particular task will be executed by a given machine. For this, every machine has an internal threshold value which is a function of different dynamic and static factors (price and time associate with task execution, machine current state, etc.). Each task to process will have its own stimuli value that will be compared with the threshold of the machines and will be used to decide which machine will perform the task. Thresholds are allowed to change and become heterogeneous over time as a function of stimuli encountered and tasks performed, and this can lead to specialization and division of labour.

14.6 Concept Validation in Prototypes

We consider some general use cases to illustrate of shop-floors with ERP systems. Status of machines on the shop floor is usually monitored. This information could represent completion of an order, number of work pieces produced or even failure of a machine, which is vital for consecutive production plans.

The main common characteristics we focus on are:

- *Smart devices*: shop-floor machines and other sensors, PLCs, and IT devices are the actors forming an “internet of things” in the factory. They all offer their functions (e.g. start/stop, swap to manual/automatic mode) or status (e.g. power consumption, mode of operation, usage statistics, etc.) through web service interfaces, either directly or through the use of gateways or mediators.
- *Business logic and visualization services*: In our prototypes, the business logic services are supported by a service composition engine and a graphical user interface (GUI) using a visualization toolkit. An operator can use existing tools to create the business rules. Via visualization tools the plant-floor status and the

overall process execution can be analysed in detail. As an example, the operator can instantiate and use a set of widgets such as gauges and graphs to monitor in real-time the status of production machines and associate it with the respective orders or business effects.

- *Enterprise applications*: This refers to high-end business software such as ERP or product life cycle management. The idea at this level is to visualize processes rather than the machines executing the processes. This layer is connected to the plant-floor devices through the other layers. As such it can report machine failures and plant-floor information on the process visualization and workflow. Furthermore, business actions (e.g. inform customers about a possible delay) can be executed based on this timely information.

14.6.1 Machine Monitoring, Dynamic Decision and Order Adaptation

On a production line we need to monitor a robotic gripper for overheating as this would cause further malfunctions. As shown in Figure 14.8, a SunSPOT wireless sensor node is attached to the gripper and checks continuously the temperature at the relevant location right before the gripper starts its operation, e.g., open or close.

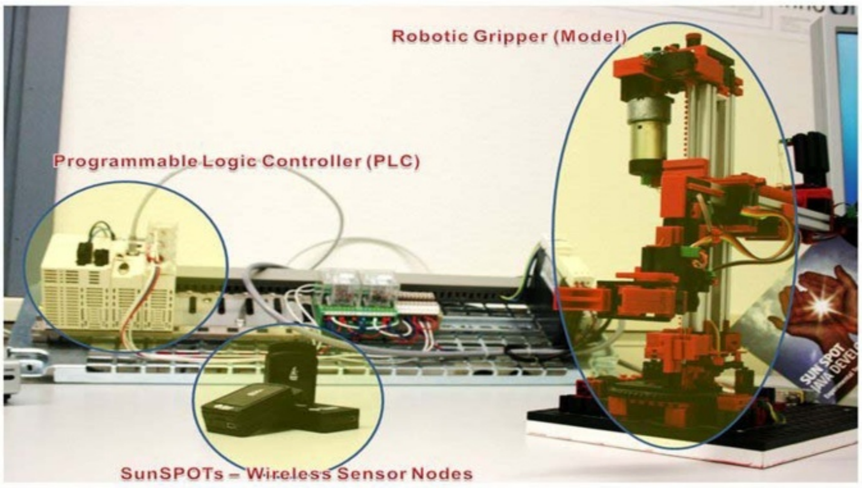

Robotic Gripper (Model)
Programmable Logic Controller (PLC)
SunSPOTs – Wireless Sensor Nodes

Fig. 14.8 Overheating monitoring via wireless sensor and ERP-supported control of a process

A PLC controls the robotic gripper and offers its available functionality as several web services; the same holds true for the wireless sensor. In SAP MII we have modelled the business logic (shown in Figure 14.9), which takes decisions based on the data from the robotic gripper and the wireless sensor. The operation of both devices generates events, which are picked up and consumed by SAP MII as we extended the tool to subscribe to these.

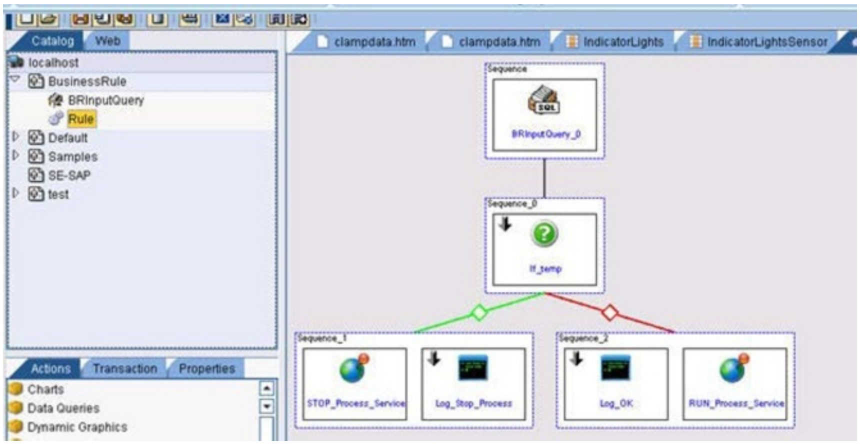


Fig. 14.9 Modelling business logic in SAP MII

In our scenario, during normal operation the robot gripper should not exceed a specific temperature limit. If the threshold is exceeded, a business rule triggers necessary countermeasures, e.g., it stops the gripper in order to prevent damage and invokes appropriate enterprise-level services. This includes visualizing the stopping of the gripper and possible delays in production, e.g. by changing the colour in the management view of the factory as depicted in Figure 14.10. If the resulting order delay is critical, a notification is generated for the key account manager, about the fact that an order for his client is in danger of missing the deadline. The timely information about production delays has the potential to prevent lost sales, which are more likely to occur if customers do not receive their order in time and are not informed promptly and reliably about it. The management cockpit of Figure 14.11 is another example of web service composition at a higher layer where we integrate the real-time status of the factory with Google Maps in order to visualize the overall effect of a single failure that results from an order delay.

Apart from the high-level view for the manager, there is also other visualization information for the operator. A simple gauge fed with the temperature data provided by the wireless sensor can be depicted also in real-time via the SAP MII tool. For this purpose, he uses manufacturing intelligence software and displays the gauge on a screen situated close to the robot. Finally, the sales manager can also benefit from the SOA paradigm, as, e.g., the output of the business rule is connected to an ERP system, which provides up-to-date information about the execution of the current orders. Whenever the process is stopped because the rule was triggered, an event is sent to the ERP system through its web service interface. The ERP system then updates the orders accordingly and informs the clients of a possible delay in the delivery.

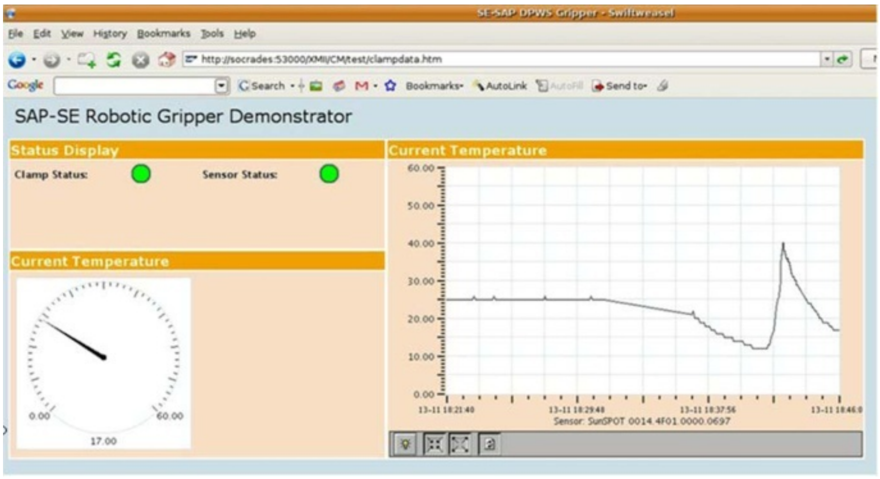
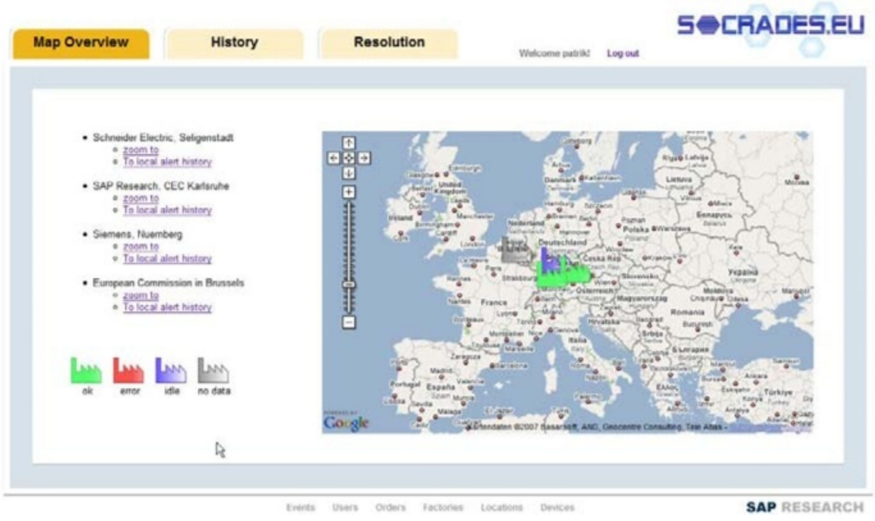

SE-SAP DPWS Gripper - Swiftweasel
File Edit View History Bookmarks Tools Help
http://socrades:53000/XMII/CM/test/clampdata.htm
SAP-SE Robotic Gripper Demonstrator
Status Display
Clamp Status:
Sensor Status:
Current Temperature
0.00
60.00
17.00
Current Temperature
60.00
50.00
40.00
30.00
20.00
10.00
0.00
13-11 18:21:40
13-11 18:29:48
13-11 18:37:56
Sensor: SunSPOT 0014.4F01.0000.0697

Fig. 14.10 Live reporting in SAP MII of the current shop-floor status based on events received from the devices



- Schneider Electric, Seligenstadt
  - zoom to
  - To local alert history
- SAP Research, CEC Karlsruhe
  - zoom to
  - To local alert history
- Siemens, Nuernberg
  - zoom to
  - To local alert history
- European Commission in Brussels
  - zoom to
  - To local alert history

Fig. 14.11 High-level factory view in the management cockpit

14.6.2 The Future Shop Floor: Mashup of Heterogeneous Service-oriented-architecture Devices and Services

Today, integrating devices in applications requires not only advanced knowledge of the device, its configuration and the way it connects, but also the installation of highly specialized software that glues the data (often in proprietary format) with applications. Such an integration model is costly, application specific and creates isolated islands for each shop floor. As a result, it is extremely hard for enterprise service developers to enrich service functionality with real-time data coming from the shop floor. The SOA concept has proven very successful for gluing heterogeneous systems, and if the same would be applicable for devices this would be a significant step forward in the direction of coupling the real-world and the business world.

To demonstrate the concept we have taken several devices (some of them IP-enabled) and wrapped their functionality with web services. More specifically as depicted in Figure 14.12, we have:

- An RFID reader: RFID tags as they appear are read by the reader, which raises events showing info with respect to the tag. Each tag is considered to be integrated with a product and serves as a token that links it with the business information (e.g. order).
- A robotic arm: the functionality of the robotic arm controller has been wrapped, e.g. grab or move.
- An IP electricity switch: an alarm lamp has been attached in an IP electricity switch, which offers the on/off functionality as a web service.
- A wireless sensor-controlled emergency button: in the IO pins of the SunSPOT wireless sensor, the emergency button has been attached. The press or release of it is captured by the SunSPOT, and a web service event is generated.
- A wireless sensor for vibration monitoring: the capability of SunSPOTs to measure acceleration is used for vibration monitoring via the sensor mounted on the robotic arm. This monitors the transportation conditions of the product by the robot to make sure it adheres to the quality guidelines for the specific product.

Production parts that arrive on the factory shop floor are equipped with RFID tags. As they reach the processing point, their data is read by an RFID reader. An event is generated by the reader with all the necessary data, e.g. RFID number, and placed on the network. The robot gets notified by this event and picks up the production part. By matching data from the enterprise system and the RFID tag, it knows how to further process the part.

In parallel, the SunSPOT mounted on the robot monitors the vibration and if this exceeds a specific threshold an event is raised to immediately stop the process. The same holds true for the emergency switch, if an operator at the shop floor for any reason wants to immediately stop the process, s/he presses this switch. The result is captured by the wireless sensor and an event is raised to immediately stop the robot.

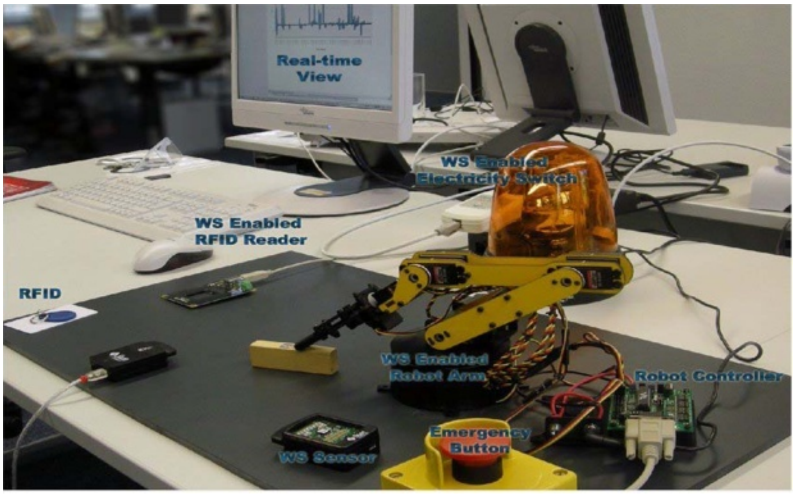
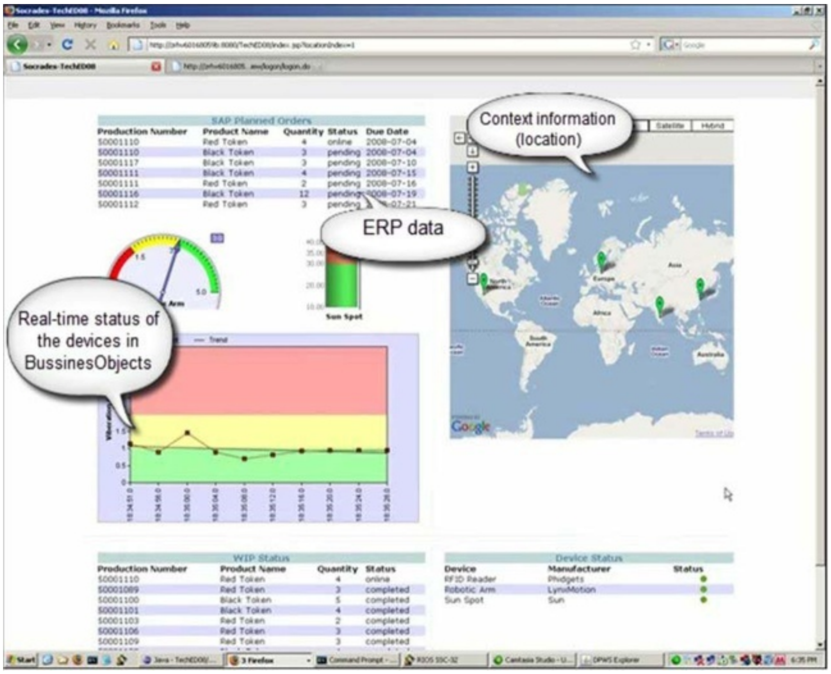


Fig. 14.12 Web-service-enhanced/wrapped heterogeneous devices - RFID readers, sensors, robot, IP electrical switch, etc

Once an event occurs, the devices process it and react accordingly. As such, the robot picks the emergency shutdown event and immediately stops its action. Also the IP electricity switch turns on the emergency light once it receives the event. At a higher level, there is an enterprise application for shop-floor reporting called SAP Business Objects. There we have a real-time monitoring of the shop-floor status as the application also subscribes to all events raised. The plant manager can immediately see (Figure 14.13) the status of the ERP orders, the production progress, and the device status, and have a global view of all factories and the possible side-effects of a production line delay due to shop-floor device malfunctions.

All of the communication is done via web service technologies and the realization of this specific scenario was made possible by having a composition of the available functionality that all devices and applications expose as a service. In legacy systems, integration of a new device or reassignment of its role would result in reconsidering how the device integrates with other devices and how they control it. However, with the SOA approach described, a new scenario is possible by modifying the orchestration of the services already available.


Context information (location)
ERP data
Real-time status of the devices in BussinesObjects

SAP Planned Orders

Production Number	Product Name	Quantity	Status	Due Date
50001110	Red Token	4	online	2008-07-04
50001110	Black Token	3	pending	2008-07-04
50001117	Black Token	3	pending	2008-07-10
50001111	Black Token	4	pending	2008-07-15
50001111	Red Token	2	pending	2008-07-16
50001116	Black Token	12	pending	2008-07-19
50001112	Red Token	3	pending	[illegible]-07-21

WIP Status

Production Number	Product Name	Quantity	Status
50001110	Red Token	4	online
50001089	Red Token	3	completed
50001100	Black Token	5	completed
50001101	Black Token	4	completed
50001103	Red Token	2	completed
50001106	Red Token	3	completed
50001109	Red Token	3	completed

Device Status

Device	Manufacturer	Status
RFID Reader	Phidgets	●
Robotic Arm	LynxMotion	●
Sun Spot	Sun	●

Fig. 14.13 Real-time reporting via enterprise visualization tools

14.6.3 Dynamic Supply Chain Management Adaptation

The SAP supply chain management (SCM) production planner (PP) system supports manufacturing by optimizing throughput times and bottleneck capacities using new scheduling processes (supply optimization) and by achieving online integration of production planning and control activities. This provides a transparent overview of the entire order network. For an optimized schedule in the production plan, the SAP SCM PP needs data about the available resources from the shop floor to cross boundary locations.

The quality and validity of the information is important for real-time simulation of the shop-floor resources. To achieve this, SAP has added MII as an integration middleware between the MES on the shop-floor and the SAP ERP systems. MII also offers bidirectional connectivity between the control, field-level systems and the ERP landscape. As a result, the plant manager could have an overview of actual orders getting executed on a particular machine and what is the status of that machine. Moreover intelligent information can be extracted from manufacturing analytics and performance factors provided by the SAP MII.

In this prototype we have used a test rig that consists of a suite of pneumatic-based hardware that does operations like picking, placing, moving, drilling, proximity sensing and stocking. The overview of the hardware is pictured in Figure 14.14. The test rig is supported by a 10-bar compressor and operates at 24 V DC from the power supply unit. It is assumed that it produces tokens, which are drilled with a hole in the centre. The tokens are supplied from a magazine. The rig drills a hole on the token, checks its colour and material, and sorts it to the corresponding storage magazine.

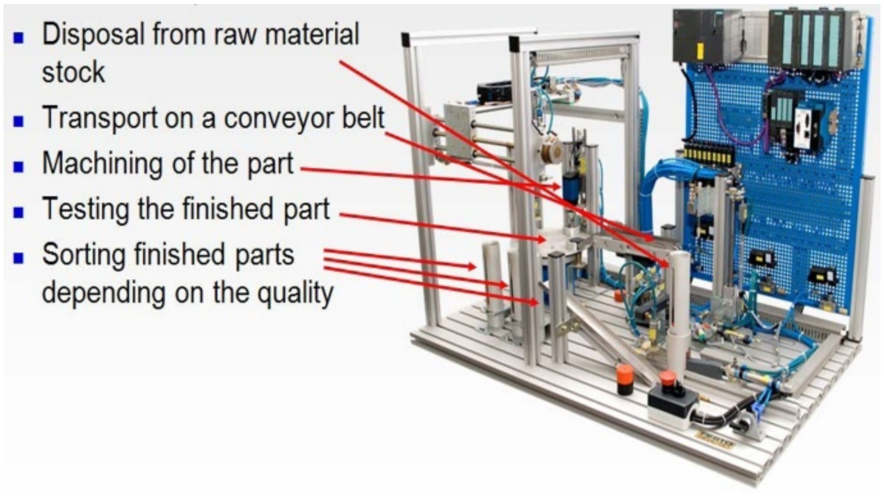

Disposal from raw material stock
Transport on a conveyor belt
Machining of the part
Testing the finished part
Sorting finished parts depending on the quality

Fig. 14.14 Main operations

The main parts of the test rig (also seen at Figure 14.15) are:

- Input magazine and distributor: this module consists of a shoving-out cylinder operated pneumatically, a pile magazine, which stocks the red and black tokens and a swivel arm.
- Proofer: the proofer module consists of the proof station, a rejection cylinder and a conveyor.
- Rotating work table: this module consists of a turn table, a drill press with clamping cylinder and a piston that checks for a hole in the token called the proof cylinder.
- Dispatcher: the dispatcher reads the colour information from the PLC and calculates the trajectory of the vertical and horizontal motion of the linear actuators, picks up the component and moves the actuators towards the storage magazine where the tokens are stored respective to their colour.

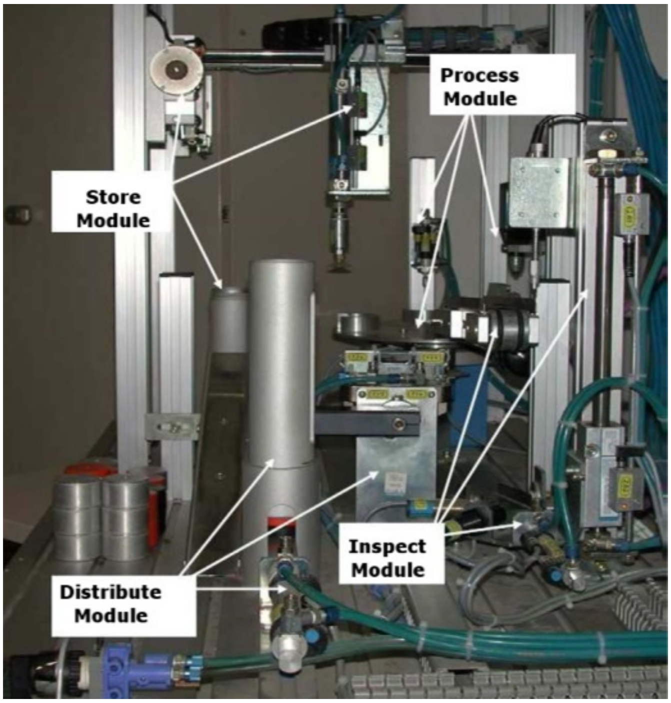
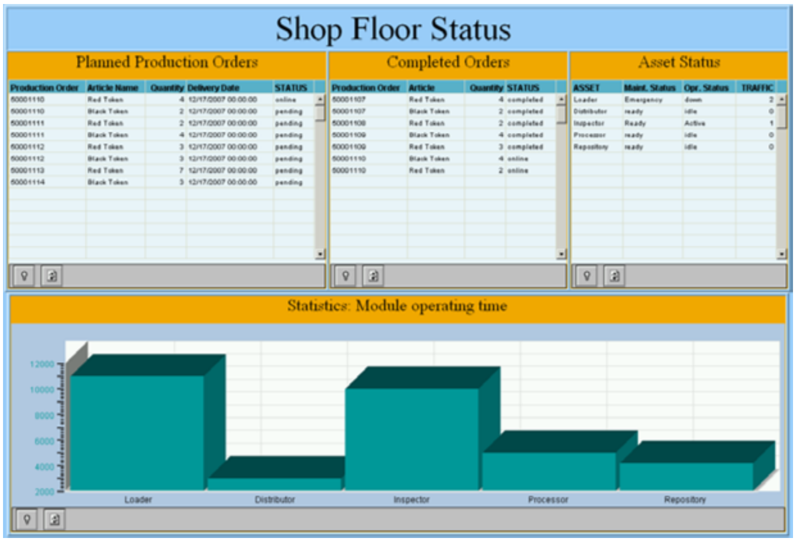


Fig. 14.15 Material flow

All these modules are controlled by one PLC. The rig uses the Siemens S7-300 PLC with an OPC DA interface to communicate to an execution system. In this demonstrator the execution operations on the PLC are wrapped with web services in a middleware component called the mediator. The mediator monitors data within the Siemens S7-300 PLC using the OPC interface. It also controls the start and stop operations of the PLC. A Mediator exposes the start and stop operations as a web service. These operations can be invoked from the SAP MII software. The mediator retrieves the status information from the PLC. It propagates this information as web service events through DPWS and is also visible via enterprise applications such as the SAP MII. At the end of each operation, there are events generated by the mediator to SAP MII. Either the successful completion of the operation or the failure of a task is reported. To that extent, we have used DPWS to connect the enterprise software and test rig via the mediator.

The SAP MII software performs the basic interconnection between the shop floor and the ERP landscape. SAP MII transactions analyse the results using complex business rules, which are developed using the MII business logic editor. The rules check which token (red or black) is produced and check against the corresponding production order. If there are more red tokens produced than the ones required in the production order, they are stored and then logically deducted in the next production plan, which is already designed and ready to be executed, and in the pipeline. This does not increase the production time, rather neutralizes the delay. The MII also shows the status of the test rig modules. Figure 14.16 presents the snapshot of the SAP MII GUI that is visible to the plant-floor manager.



Shop Floor Status

Planned Production Orders

Production Order	Article Name	Quantity	Delivery Date	STATUS
50001110	Red Token	4	12/17/2007 00:00:00	online
50001110	Black Token	2	12/17/2007 00:00:00	pending
50001111	Red Token	2	12/17/2007 00:00:00	pending
50001111	Black Token	4	12/17/2007 00:00:00	pending
50001112	Red Token	3	12/17/2007 00:00:00	pending
50001112	Black Token	3	12/17/2007 00:00:00	pending
50001113	Red Token	7	12/17/2007 00:00:00	pending
50001114	Black Token	3	12/17/2007 00:00:00	pending

Completed Orders

Production Order	Article	Quantity	STATUS
50001107	Red Token	4	completed
50001107	Black Token	2	completed
50001108	Red Token	2	completed
50001109	Black Token	4	completed
50001109	Red Token	3	completed
50001110	Black Token	4	online
50001110	Red Token	2	online

Asset Status

ASSET	Maint. Status	Opr. Status	TRAFFIC
Loader	Emergency	down	2
Distributor	ready	idle	0
Inspector	Ready	Active	1
Processor	ready	idle	0
Repository	ready	idle	0

Statistics: Module operating time

Fig. 14.16 Plant manager's view based on SAP MII

14.6.4 Taming Protocol Heterogeneity for Enterprise Services

When enterprise application designers develop new features, taming the heterogeneity can be a significant burden. The SIA architecture offers a pluggable framework with the aim to abstract the functionality from the protocol itself, at least for business-related messages that are valuable to the enterprise system. As such, business applications can be more flexibly developed, while the translation of messages for a specific protocol is pushed down to the device level.

In this prototype we considered a solar cell manufacturing plant which produces thin-film photovoltaic cells. Manufacturing such semiconductor devices is similar to producing integrated circuit chips. Semiconductor manufacturing involves complex and machine intensive workflow processes. This means, machines from different manufacturers would need to exchange large amounts of data intensively to complete a production process.

Since its specification, the SEMI Equipment Communications Standards (SECS) has been widely used on the shop floor in a semiconductor fabric (Cimetrix, 2009). SECS-II/HSMS offers the possibility of delivering SECS messages over IP, but only between two points; no multi-point communication is possible-a drawback for most internet-technology-based applications. The thin films are transported between the processing machines using conveyor belts. We consider such a conveyor system that is controlled by an OPC UA-based PLC (e.g. Beckhoff CX1010). OPC UA-based manufacturing devices are common on the shop floor like this one. We take the example a step further by monitoring the energy consumed by the whole shop floor and estimate the energy consumed to produce one photovoltaic cell. We add a smart meter to the production facility, periodically record the energy consumption data, and associate the energy costs from an ERP system. The smart meter would send data using REST-based HTTP packets (see Figure 14.17) to the enterprise application.

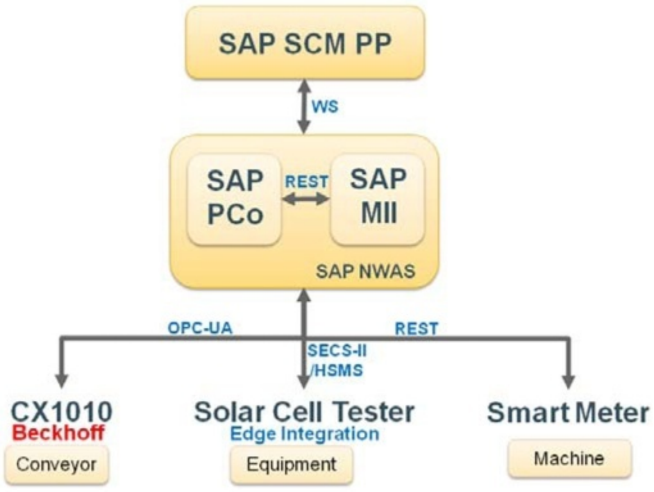


Fig. 14.17 Multi-protocol interaction with web services, REST, OPC-UA and SECS-II/HSMS

Traditionally SECS-II/HSMS messages are sent only from the equipment to the host interface, which is the MES system. Providing alternative connections or multiple communication endpoints is rather expensive and requires additional software development that can be tedious to maintain when the features of the manufacturing equipment changes. When two entities are trying to connect to SECS host equipment, the connection is terminated to either one of them for a brief period; if not, multi-point communication is supported. As a consequence, challenges arise when trying to make the processes adaptive or trying to extend it to additional destinations. Introducing variables to adapt to the dynamic nature of the shop floor is very expensive for companies that span multiple production locations and several heterogeneous IT systems. We propose to solve this problem by extending the SECS-II/HSMS with web services. Combining web services with SECSII/HSMS enables semiconductor manufacturing equipment to report alarms, events and control signals to a variety of IT systems in the plant directly in real time, avoiding expensive gateway connectors.

With new and powerful devices on the shop floor, manufacturing equipment can deliver real-time data to multiple destinations. As a result, a mashup of real-time data analysis can be done that can lead to estimation of trends from current production. In this prototype, the energy consumed to produce a photovoltaic cell can be estimated using web services on traditional protocols and REST-based light weight frameworks. Simultaneously, standard XML-based technologies like OPC UA can be mashed up from the device level, offering a new level of energy-aware production.

14.6.5 Energy Monitoring and Control via Representational State Transfer

Interacting with devices in many cases should be done very easily, in a short time and with no strict guarantees or need for a full-blown functionality. As such, heavyweight web service integration in specific scenarios might not be the best option; on the contrary, REST integration is a better match for lightweight integration. In most early web-based approaches, HTTP is used only to transport data between devices, while in fact HTTP is an application protocol. Projects that specifically focus on re-using the founding principles of the web as an application protocol are still not common. Creation of devices that are web-enabled *by design*, would facilitate the integration of physical devices with other content on the web. As pointed out in Guinard and Trifa (2009) and Wilde (2007), REST-enabled devices would not require any additional API or descriptions of resources/functions.

The architectural principle that lies at the heart of the web, namely REST as defined by Roy Fielding (Fielding and Taylor, 2002), shares a similar goal with more well-known integration techniques such as WS-* web services (SOAP, WSDL, etc.) or DPWS, which is to increase interoperability for a looser coupling between the parts of distributed applications. However, the goal of REST is to achieve this in a more lightweight and simpler manner, and focuses on resources, and not functions as is the case with WS -* web services. In particular, REST uses the web as an application platform and fully leverages all the features inherent to HTTP such as authentication, authorization, encryption, compression, and caching. This way, REST brings services "on the web, directly from the browser": resources can be linked and bookmarked and the results are visible with any web browser, with no need to generate complex source code out of WSDL files to be able to interact with the service.

Smart gateways can be used to integrate or abstract from the protocol heterogeneity. These implement in principle the lowest layers defined in SIA. As an example we have created a gateway that runs a web server, and understands the proprietary protocols of different devices that are connected to it through the use of dedicated drivers. As an example, consider a request to a sensor node coming from the Web trough the RESTful API. The gateway maps this request to a request in the proprietary API of the node and transmits it using the communication protocol the sensor node understands (e.g. Zigbee www.zigbee.org). A smart gateway can support several types of devices through a driver architecture as shown in Figure 14.18, where the gateway supports three types of devices and their corresponding communication protocols. Technical details of the Smart Gateways can be found in Guinard and Trifa (2009) and Trifa *et al.* (2009).

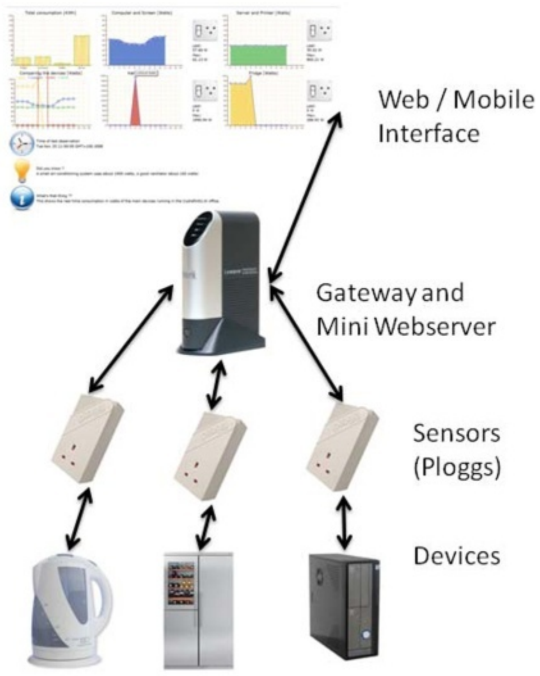


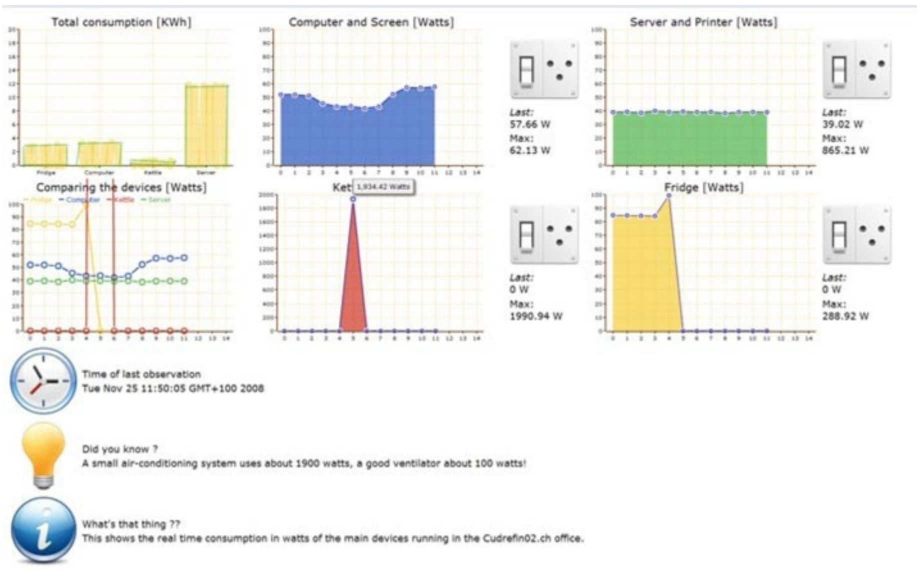
Fig. 14.18 Device integration via a gateway

In order to empirically analyse and test the potential of the RESTful approach for real-world services and how our approach could become the basis for the web of things, we implemented the architecture on two WSNs platforms: the SunSPOT sensor nodes (www.sunspotworld.com) and the Ploggs energy eensors (www.plogginternational.com). The Sun SPOT platform is a wireless sensor node particularly suitable for rapid prototyping of WSNs applications. SunSPOTs run a small-footprint Java Virtual Machine that enables the nodes to be programmed using the high-level Java programming language (Java Micro Edition CLDC java.sun.com/javame). The RESTful architecture we designed and implemented for the Sun SPOTs is composed of two main parts: a software stack embedded on each node, and a proxy server to forward the HTTP requests from the web to the SPOTs. The Ploggs smart gateway is a component written in C++ whose role is to automatically find all the Ploggs in the environment and make them available as web resources. The gateway first discovers the Ploggs on a regular basis by scanning the environment for Bluetooth devices. It then filters the identified devices according to their Bluetooth identifier. The next step is to make their functionalities available though simple URLs, and for that a small-footprint web server is used to enable access to the sensors' functionalities over the web. This is done by mapping URLs to native requests on the Plogg Bluetooth API. Hence, requesting the energy consumption of a Plogg can be done simply be issuing a GET request on the following URL, as suggested by the REST architectural principles (Fielding and Taylor, 2002): to http://webofthings.com/energymonitor/ploggs/conveyerbelt. This returns a JSON (JavaScript Object Notation) json.org/ document containing information about the energy consumption of the conveyor belt. Note that JSON is an (compatible) alternative to XML often used as a data exchange format for web mashups. Since JSON is a lightweight format we believe it quite adapted to devices with limited capabilities.

The idea of the energy visualizer prototype built on top of the Ploggs Smart Gateway and the RESTful SunSPOT is to offer a dashboard user interface on the web to control and monitor the energy consumption at the device level. It offers six real-time and interactive graphs. The four graphs on the right side provide detailed information about the current consumption of all the appliances currently in the vicinity of the smart gateways. The two remaining graphs show the total consumption (kWh), and respectively a comparison (on the same scale) of all the running appliances. Finally, a switch button next to the graphs enables one to power on and off devices over the web.

This dashboard is built as a mashup that uses the RESTful Plogg API in a Google Web Toolkit (GWT) application code.google.com/webtoolkit/. The GWT is a powerful platform for building web mashups since it offers a large number of easily customizable widgets. For the graphs shown on Figure 14.19, we use the Open Flash Chart GWT Widget Library. This library offers a comprehensive set of graph widgets.

The ambient meter prototype is implemented on a RESTful SunSPOT which uses a Ploggs smart gateway for gathering the energy consumption of any place it is located in. It uses an HTTP connector we implemented in the RESTful Sun SPOTs to contact the Ploggs smart gateway. Every 5 seconds, the SunSPOT will poll the following URL using the GET method: http://localhost/energymonitor/load. The SunSPOT changes its colour according to the energy consumption of the place it is currently located in. Thus, by combining these two services, the SunSPOT is turned into an ambient meter that can assess the energy consumption of the place it is located in, demonstrating the ease of integrating.



Time of last observation
Tue Nov 25 11:50:05 GMT+100 2008

Did you know ?
A small air-conditioning system uses about 1900 watts, a good ventilator about 100 watts!

What's that thing ??
This shows the real time consumption in watts of the main devices running in the Cudrefin02.ch office.

Fig. 14.19 Energy monitoring per machine

14.7 Discussion and Future Directions

We have presented how to deploy web services on shop-floor devices to connect them to enterprise systems. With web services, enterprise systems are also able to subscribe to events and take advantage of real-time information flow for optimization of the production planning or reaction to unexpected changes. The clear advantage of pushing SOA concepts down to device level is that business application developers can design and implement new functionality in enterprises without focusing on the devices but on the services they provide. As such, another abstraction layer based on well-known and used web service standards will ease the integration of enterprise and shop-floor systems. The scenario depicted in this chapter is a proof of concept for this easier and tighter integration. The results show that the dynamic nature of the shop floor can be utilized efficiently to plan further production orders and even implement last-minute changes on the production line using real-time data (real-time reconfiguration based on the application needs). As all communication is done via web services (in our case DPWS) it is easy for other entities (whether they are services or devices) to subscribe and get the necessary info while in parallel being agnostic to the actual implementation details. This greatly increases interoperability and reduces costly integration time.

In our next steps we plan to further work on better web service integration, also with the usage of OPC UA as well as improvements regarding fulfilment of typical industrial automation-specific requirements like real-time constraints, reliability, safety, or security. Special attention will be paid to improve easy configuration of integration components like gateway or mediator based on device description technology known within industrial automation. This will pave the way for auto-configuration on the gateway and mediator level and enhance integration of legacy systems into higher-level management applications.

14.8 Conclusions and Future Work

Ubiquitous, SOA-based device integration leading to interaction of devices with enterprise services in a timely manner is an important vision that creates substantial impact both from the perspective of research and industrial application. The paradigm of services on every layer of the network will influence the structure and operation of future intranets and the future internet. Dynamic service discovery and composition will enable a new generation of applications that will more closely couple physical environments and processes with the corresponding models in business software, which are their virtual counterparts.

We have presented in this chapter the imperatives and motivations for more dynamic and flexible production lines in the factory of the future. With increasing competition – but also collaboration – around the globe, more information exchange and cross-layer communication is becoming necessary. We have introduced technologies that will be found in the future factory. They will entail more dynamic and adaptive production equipment that will closely collaborate with each other through real-world services and adjust its behaviour dynamically. We described several prototypes that demonstrate our ideas and depicted the applicability of the concept.

To continue our work, we will further investigate collaborative intelligent behaviour and improve its stability over distributed infrastructures composed of large numbers of heterogeneous networked embedded devices. Allowing for a flexible, semantic discovery of services is also a promising topic for further research, just as providing a rigid security framework for the described environments.

Acknowledgements The authors would like to thank the European Commission and the partners of the European IST FP6 project "Service-Oriented Cross-layer infRAstructure for Distributed smart Embedded devices" (SOCRADES - www.socrades.eu), for their support.

References

Ahmed R, Hall T, Wernick P et al. (2008) Software process simulation modelling: A survey of practice. J. Simul., 2:91–102

Bonabeau E, Dorigo M, Theraulaz G (1999) Swarm intelligence from natural to artificial systems. Oxford Univ. Press, New York, 1999

Chan S, Kaler C, Kuehnel T et al. (2005) Devices profile for web services. Microsoft Developers Network Library

Cianci C, Trifa V, Martinoli A (2005) Threshold-based algorithms for power-aware load balancing in sensor networks. In: Proceedings of 2005 IEEE Swarm Intelligence Symposium, IEEE-SIS, Passadena, CA, USA

Cimetrix (2009) SECS/GEM SEMI standards overview, http://www.cimetrix.com/gemintro.cfm

Colombo A W, Karnouskos S (2009) Towards the factory of the future–a service-oriented cross-layer infrastructure. ICT shaping the world, a scientific view. ETSI, John Wiley and Sons Ltd

de Souza LMS, Spiess P, Guinard D et al. (2008) Socrades: A web service based shop floor integration infrastructure. In: Proceedings of Internet of Things 2008 Conference, Zurich, Switzerland, 26–28 March

Dickerson R , Lu J, Whitehouse K (2008) Stream feeds: an abstraction for the world wide sensor web. In: Proceedings of the 1st Internet of Things Conference (IOT), Zurich, Switzerland, 26–28 March

Drytkiewicz W, Radusch I, Arbanowski S et al. (2004) A REST-based protocol for pervasive systems. In: Proceedings of the IEEE International Conference on Mobile Ad-hoc and Sensor Systems, pp. 340–348

Fielding RT (2000) Architectural styles and the design of network-based software architectures. PhD thesis, University of California, Irvine, Irvine, California, USA

Fielding RT, Taylor RN (2002) Principled design of the modern web architecture. ACM Trans. Internet Techn., 2(2):115–150

Fleisch E, Mattern F (2005) Das Internet der Dinge: Ubiquitous Computing und RFID in der Praxis: Visionen, Technologien, Anwendungen, Handlungsanleitungen. Springer

Guinard D, Trifa V (2009) Towards the Web of Things: Web Mashups for Embedded Devices. In : Proceedings of the 2nd Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009), Madrid, Spain, 20 April

Guinard D, Trifa V, Pham T et al. (2009) Towards physical mashups in the web of things. In: Proceedings of the 6th International Conference on Networked Sensing Systems (INSS 2009)

Hoyer V, Stanoesvka-Slabeva K, Janner T et al. (2008) Enterprise mashups: design principles towards the long tail of user needs. In: Proceedings of IEEE Services Computing, 2:601–602

Jammes F, Smit H (2005) Service-oriented paradigms in industrial automation. IEEE Transactions on Industrial Informatics, pp. 62–70

Karnouskos S, Baecker O, de Souza LMS et al. (2007) Integration of soa-ready networked embedded devices in enterprise systems via a cross-layered web service infrastructure. In: Proceedings of the IEEE Conference on Emerging Technologies and Factory Automation (ETFA), pp. 293–300

Kennedy P, Bapat V, Kurchina P (2008) In pursuit of the perfect plant. Evolved Technologist, ISBN 978-0978921866

Liu X, Hui Y, Sun W et al. (2007) Towards service composition based on mashup. In: Proceedings of IEEE Service Computing, pp. 332–339

Luckenbach T, Gober P, Arbanowski S et al. (2005) TinyREST–a protocol for integrating sensor networks into the internet. In: Proceedings of REALWSN

Mahnke W, Leitner SH, Damm M (2009) OPC Unified Architecture. Springer, 1st edition, ISBN 3540688986

Marin-Perianu M, Meratnia N, Havinga P et al. (2007) Decentralized enterprise systems: a multiplatform wireless sensor network approach. IEEE, Wireless Communications. ISSN 1536-1284

Pidd M, Robinson S (2007) Organising insights into simulation practice. In: Proceedings of the 2007 Winter Simulation Conference, pp. 771–775

Priyantha NB, Kansal A, Goraczko M et al. (2008) Tiny web services: design and implementation of interoperable and evolvable sensor networks. In: Proceedings of the 6th ACM conference on Embedded Network Sensor Systems, Raleigh, NC, USA, pp. 253–266

SAP (2008) Towards a European Strategy for the Future Internet, http://www.europeansoftware.org/documents/SAP\s\do5(W)P\s\do5(F)utureInternet.pdf

Trifa V, Wieland S, Guinard D et al. (2009) Design and implementation of a gateway for web-based interaction and management of embedded devices In: Proceedings of the 2nd International Workshop on Sensor Network Engineering (IWSNE'09), Marina del Rey, CA, USA

Wilde E (2007) Putting things to REST. Technical Report UCB iSchool Report 2007-015, School of Information, November, UC Berkeley

Chapter 15
Factory of the Future: A Service-oriented System of Modular, Dynamic Reconfigurable and Collaborative Systems

A.-W. Colombo[1], S. Karnouskos[2] and J.-M. Mendes[3]

Abstract Modern enterprises operate on a global scale and depend on complex business processes performed in highly distributed production systems. Business continuity needs to be guaranteed, while changes at some or many of the distributed shop-floors should happen on-the-fly without stopping the production process as a whole. Unfortunately, there are limits and barriers that hinder the business requirements in beating tackled timely in the shop-floors due to missing modularity, agile reconfigurability, collaboration and open communication among the systems. However, as the number of sophisticated networked embedded devices in the shop-floors increases, service-oriented architecture (SOA) concepts can now be pushed down from the upper information technology level to the device level, and provide a better collaboration between the business systems and the production systems. This leads to highly modular, dynamic reconfigurable factories that build a collaborative system of systems that can adapt and optimize its behaviour to achieve the business goals. The work presented in this chapter shows directions to achieve this dynamism by means of SOA introduction in all layers, increased collaboration and close coupling of production sites and enterprise systems.

Keywords Service-oriented architecture, dynamic reconfigurable systems, collaborative systems

[1] A.-W. Colombo (✉)
Schneider Electric Automation GmbH, Steinheimer Str. 117, D-63500 Seligenstadt, Germany
email: armando.colombo@de.schneider-electric.com

[2] S. Karnouskos
SAP Research, SAP AG, Vincenz-Priessnitz-Strasse 1, D-76131 Karlsruhe, Germany

[3] J. –M. Mendes
Faculty of Engineering, University of Porto, Rua Dr. Roberto Frias s/n, 4200-465 Porto, Portugal

15.1 Introduction

The business world is highly competitive, and in order to successfully tackle everyday challenges, operational managers and executives demand high dependability and wide visibility within their business process networks. The later is done usually via business key performance indicators. However, in order to provide up-to-date information and be able to react in a flexible and optimal way to changing conditions, real-time information must flow via all layers from the shop-floor up to the business process level. In that sense enterprises are moving towards service-oriented infrastructures that bring us one step closer to the vision of real-time enterprises. Applications and business processes are modelled on top of and using an enterprise-wide or even cross-enterprise service landscape. For any solution to be easily integrated in this environment, it is recommended to feature a service-based approach (Karnouskos *et al.*, 2007).

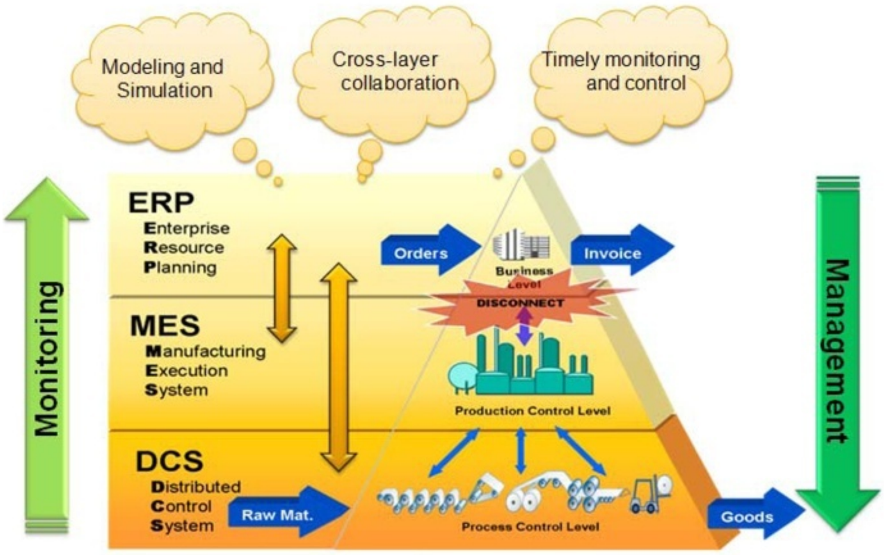


Fig. 15.1 Monitoring, control and collaboration for service-based future factories

Currently, shop-floor intelligent systems based on distributed embedded devices concentrate the programming of the behaviour and intelligence on a handful of large monolithic computing resources accompanied by large numbers of dumb devices. The intelligence and behaviour are tailored and individually programmed for each application. However, as we are moving towards the "internet of things" (Fleisch and Mattern, 2005) where millions of devices will be interconnected, provide and consume information available on the network and cooperate, new capabilities as well as challenges come into play. As these devices need to inter-operate and collaborate pursuing common production management and control goals (Colombo and Harrison, 2008), the service-oriented approach seems a promising solution, i.e. each device offers its functionality as a service, while in parallel it is possible for it to discover and invoke new functionality from other services on-demand.

The future factory should be seen as a system of systems (Jamshidi, 2008), where complex and dynamic systems interact with each other in order to achieve the goals at system-wide but also at local level. To realize this, timely monitoring and control as well as open communication and collaboration in a cross-layer fashion are key issues (Figure 15.1). Modern approaches such as the service-oriented-architecture (SOA) paradigm when applied holistically can lead to the desired result. By considering the set of intelligent system units as a conglomerate of distributed, autonomous, intelligent, proactive, fault-tolerant and reusable units, which act as a set of cooperating entities, a new dynamic infrastructure, a system of systems that is able to provide a better insight to its components to the higher levels and better react to dynamic business changes, can be realized.

15.2 The Emergence of Cooperating Objects

The rapid advances in computational and communication parts in embedded system, is paving the way towards highly sophisticated networked devices that will be able to carry out a variety of tasks not in a standalone mode as usually done today, but taking into full account dynamic and context specific information. These "objects" will be able to cooperate, share information, act as part of communities and generally be active elements of a system.

Cooperating objects (COBS) is an emerging domain (Marron *et al.*, 2009) with strong roots in (i) ubiquitous computing, (ii) embedded systems and (iii) wireless sensor networks. They possess features attributed to all three aforementioned domains, in order to allow their cooperative behaviour to emerge.

Generally COBS possess the ability and possibly the willingness to work or act together; however, their cooperation can be intentional or unintentional. Intentional cooperation can be forced (rare) or voluntary (the usual case). In a system view, COBS have goals and work together because of one or more common (even partially common) goals or means to achieve the end-goals. Single COBS are parts of teams; as such cooperative behaviour may be shown at higher level, e.g., group level, and not be clearly identifiable at object level.

The COBS are physical objects and should have the means to cooperate. This assumes:

- networking capabilities – communication;
- interaction with other objects, including heterogeneous ones; and
- autonomous behaviour.

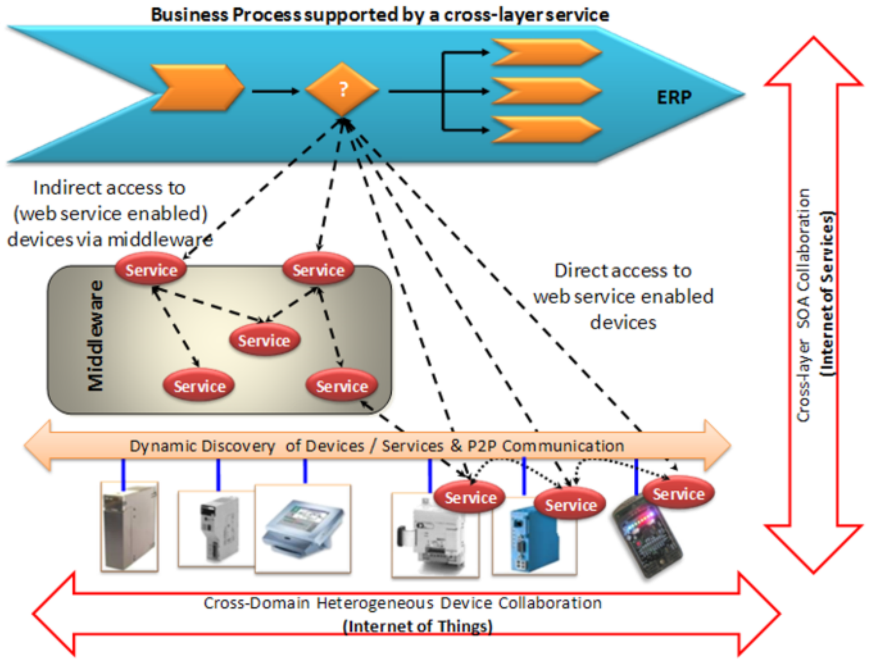

Business Process supported by a cross-layer service
?
ERP
Indirect access to (web service enabled) devices via middleware
Middleware
Service
Direct access to web service enabled devices
Dynamic Discovery of Devices / Services & P2P Communication
Cross-layer SOA Collaboration (Internet of Services)
Cross-Domain Heterogeneous Device Collaboration (Internet of Things)

Fig. 15.2 The envisioned cross-layer SOA-based vertical and horizontal collaboration

There are several flavours of COBS. Advanced COBS can process the context of cooperation intentionally, act on it and intentionally extend it, change it or stop it. As such COBS may possess logic to understand semantics and build complex behaviours. This eventually means that they can be part of dynamic complex ecosystems. Of course a cooperating object is governed by its internal rules and constrained by its resources; however, its behaviour is the result of a negotiation and potential benefit yield with respect to the external collaboration.

In our work within the SOCRADES project (www.socrades.eu), we have focused on some of the characteristics on COBS, i.e., automation devices for the future factory that have the envisioned capabilities to support web services (WS) on devices. As such their functionality is depicted as a service that can be used by other entities and eventually be part of complex orchestrations. As depicted in Figure 15.2 this will enable cross-layer collaboration not only at horizontal levels e.g., among cooperating devices but also at vertical level between systems located at different levels of a computer-integrated manufacturing or plant-wide system enterprise architecture (Pfadenhauer *et al.*, 2006; PERA, 2006; Namur, 2007). Focusing on collaboration and taking advantage of the capabilities of cooperating objects poses a challenging but also very promising change on the way future factories will operate, as well as to the way we design software and model their interactions.

15.3 The Cross-layer Service-oriented-architecture-driven Shop Floor

The key issue for achieving business continuity and realize future factories that can promptly respond to dynamic situations is the strong collaboration of the shop-floor system with enterprise and engineering systems. In order to achieve that we have to make sure that an open infrastructure (Figure 15.3) will enable the exchange of information, exposed and able to be consumed as services, and allow also the integration of legacy or isolated sub-systems.

The SOA applicability in industrial automation (Jammes and Smit, 2005) and the collaborative automation system as part of the collaborative manufacturing management (Gorbach and Nick, 2002; Nick and Polsonetti, 2003) are complementary paradigms that result from a multidisciplinary activity, including scientific and technological areas like knowledge management, production control engineering, information and communication technologies, etc. (Kennedy *et al.*, 2008; Candido *et al.*, 2009). In order to realize SOA and collaborative systems, three main activities have to be performed:

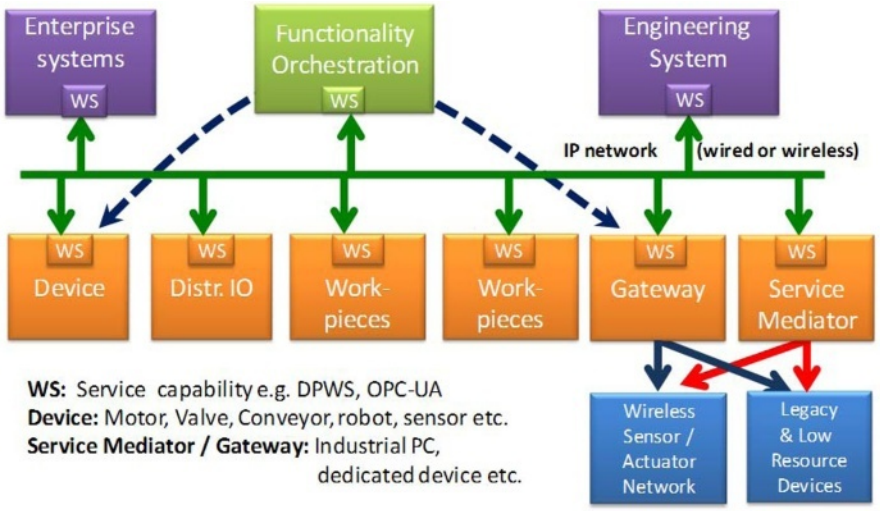

Enterprise systems
WS
Functionality Orchestration
WS
Engineering System
WS
IP network
(wired or wireless)
WS
Device
WS
Distr. IO
WS
Work-pieces
WS
Work-pieces
WS
Gateway
WS
Service Mediator
Wireless Sensor / Actuator Network
Legacy & Low Resource Devices
WS: Service capability e.g. DPWS, OPC-UA
Device: Motor, Valve, Conveyor, robot, sensor etc.
Service Mediator / Gateway: Industrial PC, dedicated device etc.

Fig. 15.3 A shop-floor seen as a cross-layer SOA-based system composed of devices, gateways, tools, engineering systems, and enterprise systems

- Identification of the cooperating entities (systems): the identification of the collaborative automation units that are able to expose and/or consume services, for each production scenario in a defined production domain, e.g., electronics assembly, manufacturing, continuous process, etc. A collaborative unit can be a simple intelligent sensor or a part/component of a modular machine, a whole machine and also a complete production system.

- Building the system of systems: networking/bringing the entities together within an SOA or collaborative infrastructure, i.e., putting the units architecturally together.
- Making the system work for reaching the production goal: collaborative behaviour of the systems for reaching common objectives, i.e., control objectives, production specifications, markets objectives, etc.

The main technical direction is to create a service-oriented ecosystem (SOA-based ecosystem): networked systems that are composed of smart embedded devices interacting with physical, engineering and organizational environments, pursuing well-defined system goals. Taking the granularity of intelligence to the device level allows intelligent behaviour to be obtained on the shop-floor of a factory by composing, aggregating and then orchestrating services offered by configurations of automation devices, engineering systems and enterprise resource planning (ERP). All these systems offer their functions as services (e.g., WS) and are able to introduce incremental fractions of the whole intelligence required for the flexible and agile behaviour of the whole system, within the enterprise/intra- and or inter-enterprise architecture (Figure15.3).

A key idea of the future SOA-based factory is to enable the participation of all factory entities including the legacy devices in the envisioned infrastructure. As of course not all devices will be able to participate, the concept of gateways and service mediators (Figure15.4) that tackle this issue has been developed (Karnouskos *et al.*, 2009).

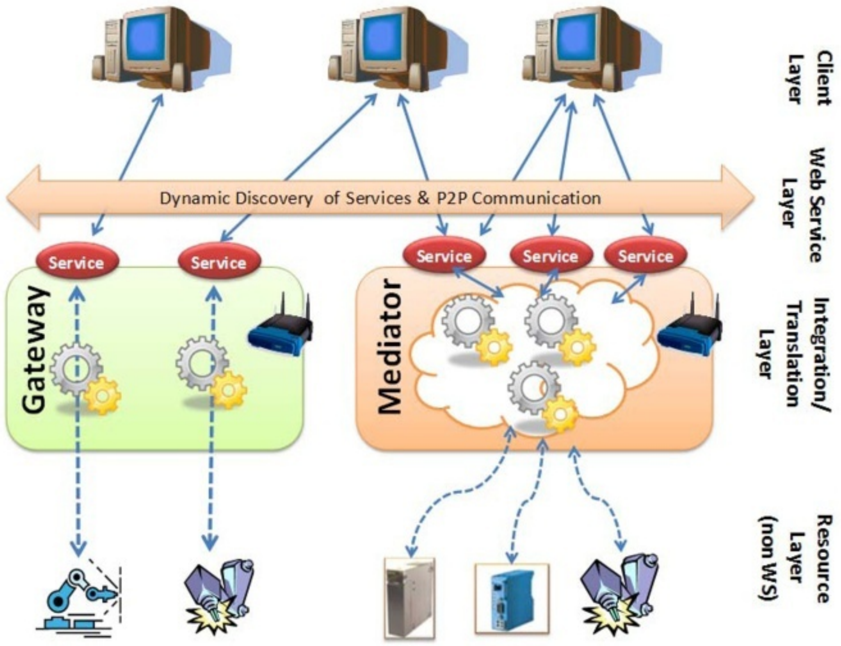


Fig. 15.4 Legacy device integration: gateway and service mediator concepts

A gateway is a device that controls a set of lower-level non-service-enabled devices, each of which is exposed by the gateway as a service-enabled device. This approach allows one to gradually replace limited-resource devices or legacy devices by natively WS-enabled devices without impacting the applications using these devices. This is possible since the same WS interface is offered this time by the WS-enabled device and not by the gateway. This approach is used when each of the controlled devices needs to be known and addressed individually by higher-level services or applications.

The service mediator not only aggregates various services but possibly also computes/processes the data it acquires before exposing it as a service. Service mediators aggregate manage and eventually represent services based on some semantics (e.g., using ontology's). In our case the service mediator could be used to aggregate various non-WS-enabled devices. This way, higher-level applications could communicate with service mediators offering WS, instead of communicating with devices with proprietary interfaces.

The holistic cross-layer SOA approach favours adaptability and rapid (production real-time conditions) reconfigurability, as re-programming of large monolithic systems is replaced by reconfiguring loosely coupled embedded units, across the complete enterprise. From a functional perspective, the main technological challenge is on managing the vastly increased number of intelligent devices and systems and mastering the associated complexity that emerges from the architectural and behavioural specifications of the factory shop-floor. In this sense, from a run-time infrastructure viewpoint, the shop-floor is now considered as a heterogeneous set of a new breed of very flexible real-time embedded devices (wired/wireless) that are fault-tolerant, reconfigurable, safe and secure, and that are exposing their functionality as a set of WS. Auto-configuration management is then addressed through basic plug-in, plug-and-work/run and plug-out mechanisms. This functional perspective must be applied to several types of devices, used in automation systems, automotive electronics, telecommunications equipment, building controls, home automation, telemetry, medical instrumentation, etc.

15.4 Dynamic Reconfiguration of a Service-oriented-architecture-based Collaborative Shop Floor

Dynamic reconfiguration of the shop-floor can be achieved by applying the concepts we introduced in engineering and operation of collaborative automation and production systems (Gorbach and Nick, 2002) composed of distributed, reconfigurable and collaborative service-oriented devices integrated in a cross-layer service-oriented enterprise architecture.

15.4.1 Methodology

The methodology we follow explains the autonomous behaviour of systems composed of service-oriented devices at the shop-floor, able to interact with the information technology (IT) enterprise systems. The resulting engineering approach permits after initial setup the automatic operation of the device and interaction to other devices and to upper IT levels of an enterprise based on, e.g., MES (manufacturing execution system)/ ERP and DMS (decision-making system).

At shop-floor level, the devices have a degree of autonomy in the sense of auto-sustainable control and necessary services to permit lateral collaboration with other devices, requesting/providing decision information from MES/ERP/DMS and integration. All interactions and resource sharing are via service orientation. There is a lose-coupling inheritance in form of bottom-up perspective (from the devices/shop-floor level), enhancing the autonomy and consequent reconfiguration capabilities.

The orchestration is described and implemented by a dynamic model-based component called the orchestrator and is supported by routines to handle undocumented events and decide over present conflict situations that arise due to shared resources, competition and concurrent relationships among the devices, as well as the occurrence of unexpected failures, errors, etc.

It is important to recall here that the operational behaviour of the devices is self-controlled and guided by internal/external events that also may correspond to service calls. Several procedures are defined for the operation behaviour life cycle of these components (see Figure 15.5):

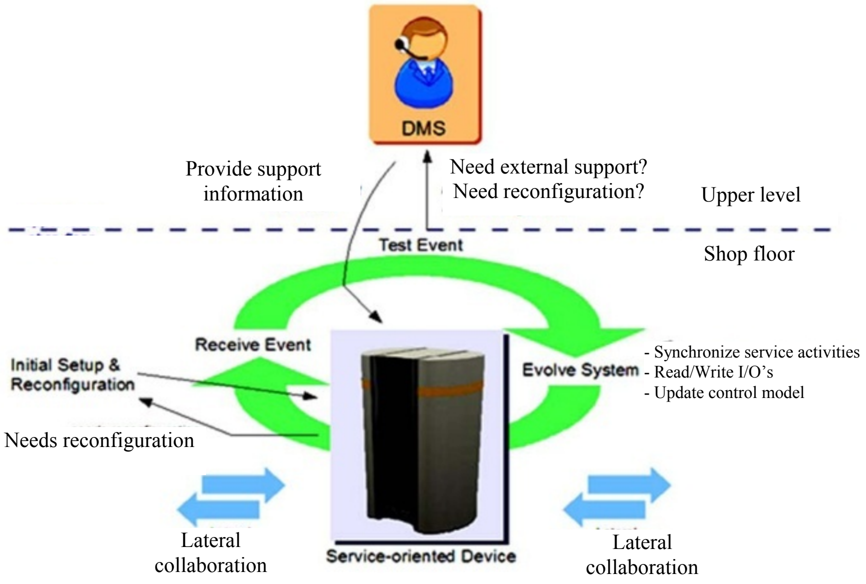


Fig.15.5 Concept of the independent behaviour of autonomous service-oriented devices/components

- Initial setup of the device, including configuration, establishment of connections to other devices/components and putting in a waiting state.
- Events are received via service operations, internal device interface to input/output (I/O) and generated directly by the control (e.g. from conflicts).
- The received event must be tested:
  - If it corresponds to some description of the control model's actual state, then the system is evolved by updating the control model, synchronizing service activities (e.g. interaction with other devices), read/write to I/O and other related tasks.
  - In the case of an exception, undocumented event or an internal conflict, some decision is required. If the device has the necessary information to resolve it, special procedures are taken to interfere in the normal system's control and new events are generated. In the case of not having sufficient control over the event, it may ask for external support (e.g., DMS) to provide more useful information for a concrete decision over the problem.
- After processing the event and evolving the system it is able to receive other events.
- Some events and consequent decisions may result in the requirement of reconfiguration of the control model and other parameters. In this situation, the device should be setup considering the new situation. Note: the reconfiguration time does disable the device, but does not imply the inability of other components/devices to operate (except on heavy dependence).

The application of this method results on autonomous devices that are self-controlled and have less dependency on other components, especially from the upper levels such the DMS. In short, the features of these devices are:

- service orientation;
- autonomous control and consequent behaviour;
- event-based life cycle.

The orchestration (control) of the lifter is defined in a high-level petri net (HLPN) model that shows the global behaviour in the different operation modes (Figure 15.6) exposed as “Services”. Please note that for simplification, only one pallet at a time may occupy the lifter. The blue-colored and larger transitions mean a complex operation (such as a service call) and can be detailed to provide a more in-depth look at the control. They represent the necessary steps to do, e.g., a top-left-transfer-in operation by reading/writing to corresponding I/O and synchronizing the service activity (e.g. a conveyor requests the service). The activation of this transition is done when the lifter is available and when a conveyor asks for the service or a sensor did detect a pallet. Other situations that are not documented can also be handled and need special procedures, as referenced previously. Transfer-out operations (such as the top-right-transfer-out transition) should be done synchronously with connected transfer devices (e.g., conveyor) to be able to provide a smooth transitional movement of the pallet from one device to another one. This requires that the lifter request a transfer in service of the connected conveyor.

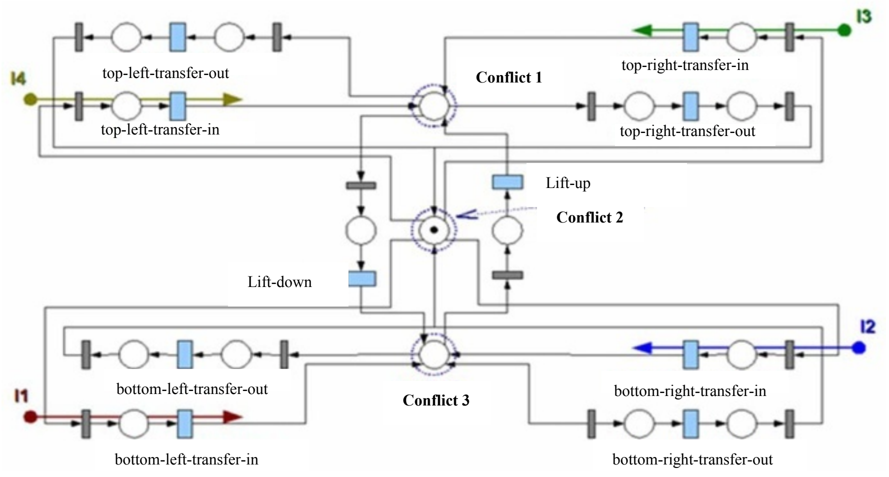


Fig. 15.6 HLPN model of the behaviour of a dynamic orchestrator for the lifter of Figure 15.5

15.4.2 Example

The methodology is applied to a mechatronics device corresponding to a lifter with two levels and four different ports where pallets can be inputted and outputted. These ports should be used to connect to other devices, such as conveyors, but can be triggered manually by placing a pallet in the lifter (since a sensor detects it). Figure 15.7 shows a representation of the lifter with all possible predicted operation modes (that correspond to the paths that a pallet may take).

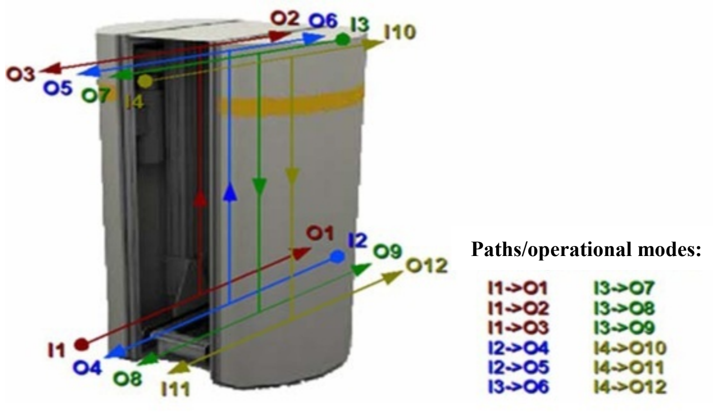


Fig. 15.7 Production device (lifter) with 12 operational modes to be exposed as services

After the initial setup and configuration of the device with the control model and additional routines, the device is available and waiting for events. For example, a connected conveyor is requesting the bottom-left-transfer-in service. In this case, and since it is a documented event in the control model, the device proceeds to evolve the system by running the HLPN model and taking the related actions. After the bottom-left-transfer-in is successfully concluded, the system has to confront an exceptional event that is thrown up by a conflict in the model (namely, conflict 2). If it does not have the necessary information to decide, it must call specialized components to help in the procedure. As depicted in Figure 15.5, DMS are used for this purpose. The lifter sends a request for support to the DMS, including supporting information (the ID of the pallet and possible outputs: lift-up and bottom-right-transfer-out). Based on the workflow of the pallet, a decision is returned in the form of an event to the lifter and now it should be able to resolve to conflict and evolve the system. For example, the lifter receives the suggestion to do a lift-up from the DMS, so it may proceed to do a lift-up. In any case, the final decision is up to the lifter that considers the received suggestion, but may operate differently in the case of internal situations (e.g. occupation of the lifter and/or connected conveyors).

The following topics summarize the initial advantages of the proposed approach:

- autonomous supervisory control and support for individual device reconfiguration without affecting whole system behaviour;
- integration into IT-enterprise and lateral collaboration among devices due to service orientation.

15.5 Analysis Behind the Engineering Methods and Tools

15.5.1 Applying Functional Analysis to Validate Service Composition Paths in High-level Petri-net-based Orchestration Models

Given a factory layout, its structure and predicted behaviour are modelled using HLPNs. A bottom-up approach is used, which consists in:

1. Modelling the behaviour of resources hardware (HW) mechatronics like robots, machines, transport components, etc. The models represent all possible discrete states of such a resource and also all manufacturing functions that this resource is able to expose as services, e.g., move-piece, pick-part, transfer-pallet, etc.
   – Remark 1: the modelling approach generates a set of a resource's discrete states that fulfil basic properties like boundedness, conservativeness, etc. (Silva and Valette, 1989; Feldmann and Colombo, 1998).

   – Remark 2: The modelling approach generates a set of a resource's exposed services that fulfil basic properties like repetitiveness, liveness, etc.

2. The models of resources are composed into a "coordination model". At this point, it is possible to speak about a behavioural model of the SOA system, which represents the set of services that are correlated and able to be orchestrated and composed/coordinated following the layout and also the production specifications of the production system.

   This task follows the same rules of configuring a required HW layout, i.e., taking into account competition, concurrency and shared resources behavioural relationships, among others. The result is an HLPN model of the whole factory.
3. Due to the strong mathematical background that is behind the Petri net theory, the models can formally be analyzed (Memmi and Rucairol, 1979; Murata, 1989).
   – Co-related discrete states of composed resources belong to, at least, one state-invariant. The set of "state-invariants" and their linear compositions represents all possible configurations of HW resources that are able to expose a service.
   – Co-related exposed services belong to, at least, one service-invariant. The set of "service-invariants" and their linear compositions represents all possible "service orchestrations paths" that the whole system is able to expose.
4. The coordination model structure represents the kernel of an SOA component identified here as model-based dynamic orchestrator.
5. When the model-based orchestrator is interacting in real-time production conditions with the HW devices, interaction based on the exchange of events and "exposition/calling" services, many known "intelligent supervisory control and automation functions" like model-based monitoring, diagnosis, maintenance control, are intrinsically supported and performed. Note: the application of Petri nets to perform monitoring of flexible production systems has been addressed by (Silva and Valette, 1989; Feldmann and Colombo, 1998 and the references therein). One of the major references to model-based monitoring systems and applications is Du *et al.* (1995).
   – The intrinsically contained information about a resource's states, exposed services, controlled manufacturing process, etc. that is contained in the HLPN model can be classified/recognized as model-based monitoring indexes.
   – The model-based monitoring indexes can/must also be exposed as services.
   – The model-based monitoring services will be used to enhance the feature-based monitoring indexes that are exposed as services by the devices. Feature-based monitoring indexes refer to sensor signals, device parameters like motor velocity, electrical intensity, etc.
6. Having the minimal set of state- and service-invariants, the dynamic orchestration that can be performed by the HLPN-based orchestrator will consist in generating and exposing different compositions of those invariants.
   – Each composition is an orchestration path, allowed by the modelled system.

   - Each composition will be done by weighting the individual atomic services. It is basically a composition of services invariants done according to pre-conditions required to automate the behaviour of the system, i.e., minimal energy-consumption service path, faster throughput service path, etc.

7. Since the system (e.g., resources and factory) and their HLPN model are mathematical (algebraic) vectorial fields (both are dual fields), the service orchestration is respecting all rules that such vectorial spaces have, under the laws of functional analysis (Kreyszig, 1989). Figure 15.8 shows a simplified example.
   - There is an isomorphism between the "composition of service-invariants" coming from the processing/analysis of the HLPN model and the "composition of services" exposed by the resources/factory.
   - The cardinality of the state and service spaces of the real production environment is the same as the places and transitions of the HLPN model.

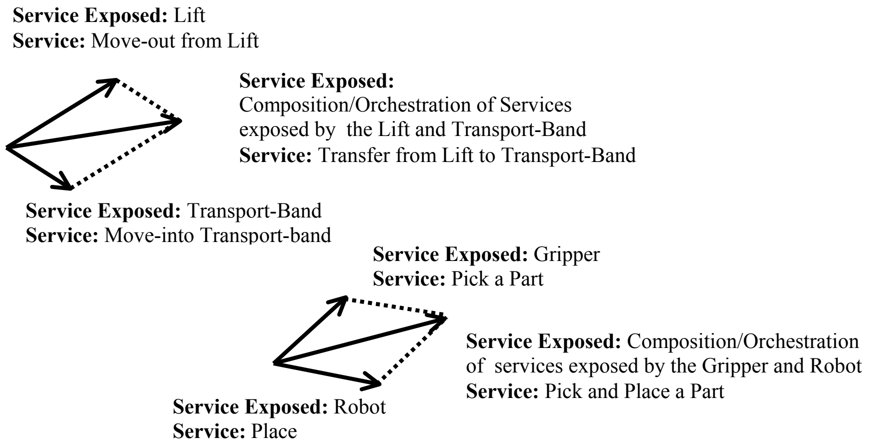

Service Exposed: Lift
Service: Move-out from Lift
Service Exposed: Composition/Orchestration of Services exposed by the Lift and Transport-Band
Service: Transfer from Lift to Transport-Band
Service Exposed: Transport-Band
Service: Move-into Transport-band
Service Exposed: Gripper
Service: Pick a Part
Service Exposed: Composition/Orchestration of services exposed by the Gripper and Robot
Service: Pick and Place a Part
Service Exposed: Robot
Service: Place

Fig. 15.8 Orchestration of services formally expressed as vectorial composition

8. The different HLPN models have to be analysed, before they can be used in the different phases of the life cycle of the SOA-based system. With the results of this analysis, a formal validation of the specification of the SOA-based architecture, structural and behavioural specifications, is performed. If and only if the basic properties of the modelled system are proved, the model can be used as the logic structure of the orchestration of services. Remark: as a matter of fact, and as an example of the high value of this analysis, only if each transition of the HLPN model belongs to at least one transition-flow/invariant, will the corresponding service at some moment be able to be exposed and/or called in the system. In a similar way of thinking, many other relationships will be validated. Figure 15.9 shows a screen-shot of the HLPN-analysis software developed in the project SOCRADES.

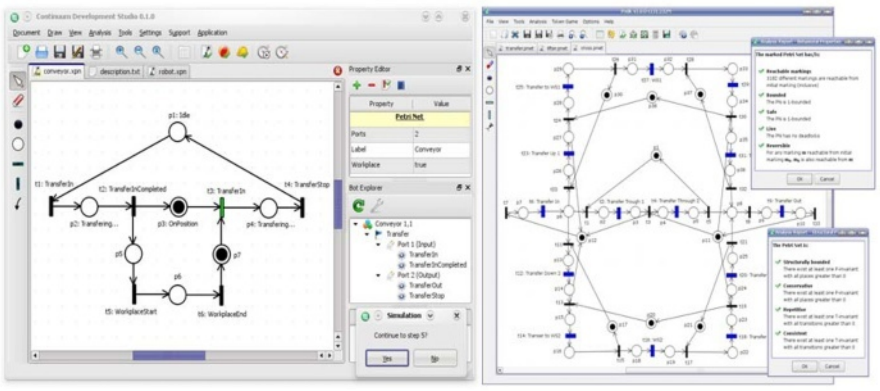


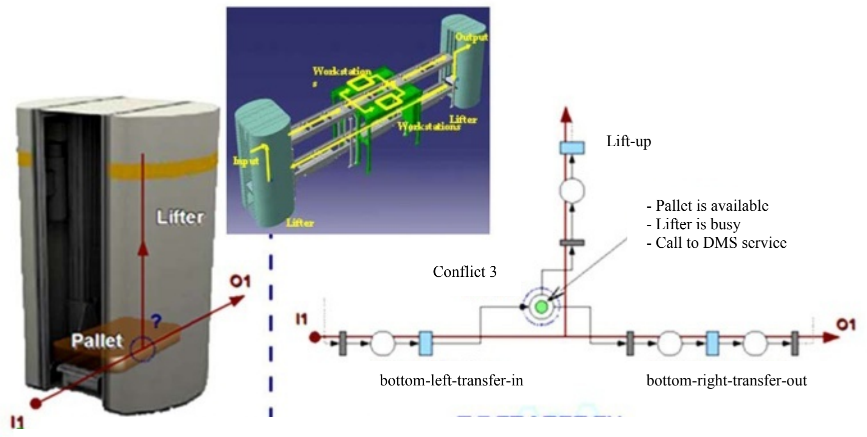
Fig. 15.9 PNDK tool to analyse the HLPN models of an SOA-based shop-floor

15.5.2 Example

1. The lifter depicted in Figure 15.7 possesses 12 operation modes.
2. Each operation mode is exposed as a service, which is the composition of atomic services.
3. Each operation mode corresponds to a transition-flow/transition-invariant or service-invariant in the HLPN model depicted in Figure 15.6.
4. The model-based orchestrator is able to expose three possible conflicts situation, i.e., states that are shared by different orchestration paths. In the language of HLPN models, the situation is recognized as "conflict" between two or more concurrently enabled transitions or transition-firing modes, as shown in Figure 15.10.
5. When the lifter is working, services are called according to an orchestrated service-based production path.
   – From control point of view, each time a service is called, it happens by the firing of a transition of the HLPN model (e.g., top-right-transfer-out in Figure 15.6).
   – A path is performed by following a precise mathematically well-defined service-invariant in the HLPN model.
   – Due to the composibility rule expressed above, at any time in an asynchronous manner (better to say, at any discrete state of the system) it is possible to take the decision for changing the next service. That is, the system follows the originally orchestrated path calling the next service of the path, or it decides to change to another service that is part of one of the minimal service-invariants that have originated the "composed path".
   – In Figure 15.6, after the system starts performing I1, after finishing the "I1-Service" and when the Petri net marking is ready to call the service

<Transfer-Up>, the service represented by O2 is no longer more exposed, e.g., blocked/failed, or the system has to change strategy due to some external conditions. The system offers the possibility to change the sequence initially orchestrated. There are three services that can follow the I1, one is O2 (out of the question), the other two are O1 and O3.

- Following the behaviour of the HLPN-based model-based orchestrator depicted in Figure 15.6, after the service <Botom-Left-Transfer-In> was executed, the conflict 3 is exposed. For the lifter, it is possible to continue with (<Lifter-Up> + <Top-right-transfer-out>) or (<Lifter-Up> + <Top-Left-Transfer-Out>) or <Botom-Right-Transfer-Out>), nevertheless, the first one according to (f.) is no longer exposed.
- The conflict situation exposed as a service in Figure 15.10 needs to be solved. The solution, i.e., the selection of the possible next-called service can be modelled in a static manner or left to an external SOA component that in the SOCRADES project has been called DMS, as depicted in Figure 15.5. Basically, the action of the DMS should be the selection of the HLPN transition or transition-firing mode that has to be fired in the next event (from the set of them that are enabled).
- This selection corresponds to a dynamic change of T-flows and it is what is called here dynamic orchestration.



- **Mechatronic transport device offers transport functions**
- **Operations are exposed as Web Service (Transfer Interface)**
  - TransferIn (int p), TransferOut (int p), TransferStop (), Transfer Completed (Pallet, p), GetStatus (Status s), …
- **Device offers additionally DPWS standard plug'n play functions for description and discovery of Device/Transfer Service**

Fig. 15.10 Conflict behaviour processing by the dynamic orchestrator for the lifter of Figure 15.7

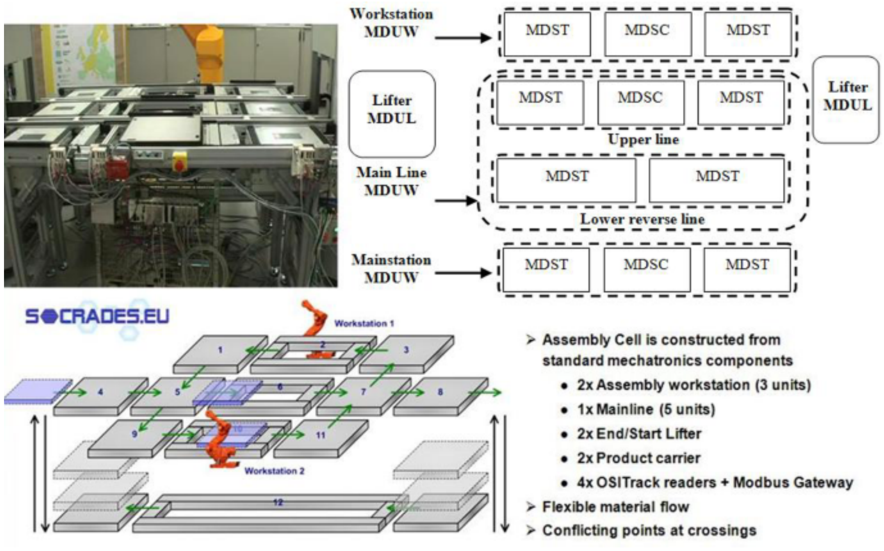
6. From control and monitoring viewpoints, the call of a service/firing of a transition is one step (service) of an orchestration path, i.e., it is possible to immedi-

ately know how many and which are the remaining steps (services) that can be called as next in the current path and also in alternative paths.

7. At any state of the system (represented by the current marking of the HLPN), past, current and future discrete states of resources (individual or composed services) are known and ready to be used for monitoring and other supervisory control functions.

15.6 A Service-oriented-architecture-based Collaborative Production Management and Control System Engineering Application

The approach for creating complex, flexible and reconfigurable production systems is that these systems are composed of modular, reusable and collaborative components that expose their production capabilities as a set of services. This composition approach applies to most levels of the factory floor, where simple devices compose complex devices or machines, which in turn are composed to build cells or lines of a production system and so on (see Figure 15.11).



- Assembly Cell is constructed from standard mechatronics components
  - 2x Assembly workstation (3 units)
  - 1x Mainline (5 units)
  - 2x End/Start Lifter
  - 2x Product carrier
  - 4x OSITrack readers + Modbus Gateway
- Flexible material flow
- Conflicting points at crossings

Fig. 15.11 A modular, reconfigurable shop-floor

The same applies to the concept of service-oriented production systems, where the shop-floor components expose their production capabilities as a set of services. Figure 15.12 shows the same layout of Figure 15.11 but identifying the shop-floor components by their "service exposition". Moreover, some of the shop-floor components are also able to expose complex services built by composing and/or orchestrating simpler services, using their collaborative behaviour.

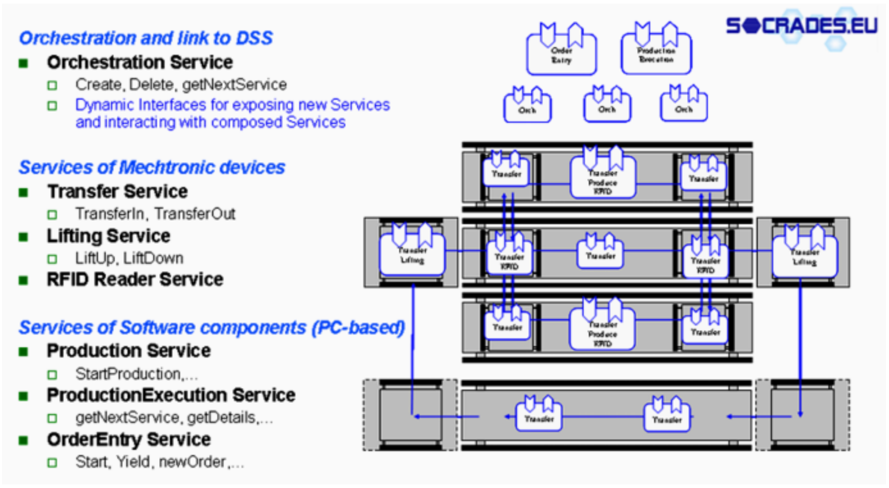


Fig. 15.12 A service-oriented view of a shop-floor

Based on the requirements of both physical hardware and the SOCRADES project, it was decided that the basic building blocks that compose the distributed system should be configurable software components assuming different tasks. Therefore, the software components were designated as "bots" (that have a so-called "orchestration engine" embedded inside) and are able (in a service-oriented fashion) to coordinate their activity and proceed also to collaboration processes with other components in the system. To design, configure and maintain bots, there is a need of specific tools that are user-friendly and speed-up the development, using a high-level programming approach (visual languages). These tools are addressed here under the name "Continuum"as depicted in Figure 15.13 (Mendes *et al.*, 2009).

Since services aren't isolated entities exposed by the intervenient software components, there should be some kind of logic that is responsible for the interaction. The modelling language of choice derives from Petri net specifications i.e., HLPN, with high-level extensions, such as time considerations, property system and customizable token game engine. The extensible property mechanism of the HLPN is used as the interface for the configuration of Web service related properties.

Following the approach and structure shown in Figure 15.5, additional requirements are the use of decision support system (DSS) and DMS (Leitao *et al.*, 2008) that are able to provide the correct information each time there are decision points in the executed Petri net model. This DSS is the main interface between the model-based approach for the shop-floor system and the production planning system.

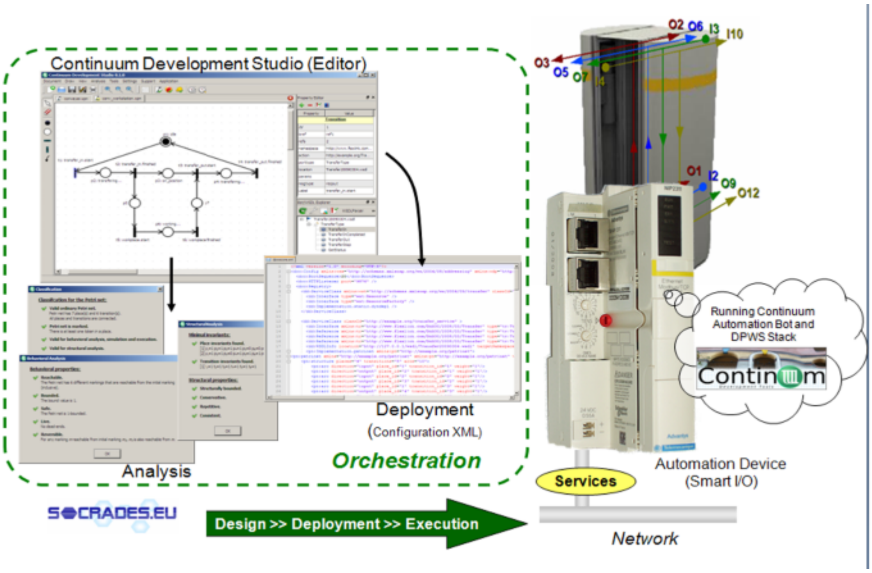

Continuum Development Studio (Editor)
Analysis
Deployment
(Configuration XML)
Orchestration
S●CRADES.EU
Design >> Deployment >> Execution
Running Continuum Automation Bot and DPWS Stack
Continuum
Automation Device (Smart I/O)
Services
Network
O2 O6 I3 I10
O3 O5 O7 I4
O1 I2 O9 O12

Fig.15.13 Petri-net-based orchestration tools and engineering continuum.

Once the models are specified for the mechatronic components or even bigger systems by the Petri net designer, tools are needed for composing systems, creating configuration files and deploying those files to simulators or even embedding the engines and files into real smart automation and control devices. Figure 15.14 shows the case of embedding the orchestration engines and deploying the orchestration models to smart I/O devices (Advantys STB from Schneider Electric).

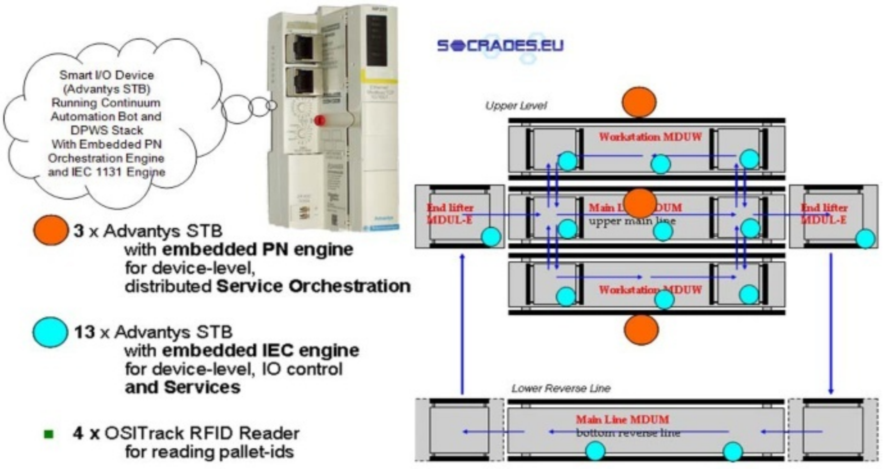

Smart I/O Device (Advantys STB) Running Continuum Automation Bot and DPWS Stack With Embedded PN Orchestration Engine and IEC 1131 Engine
3 x Advantys STB with embedded PN engine for device-level, distributed Service Orchestration
13 x Advantys STB with embedded IEC engine for device-level, IO control and Services
4 x OSITrack RFID Reader for reading pallet-ids
S●CRADES.EU
Upper Level
Workstation MDUW
End lifter MDUL-E
Main Line MDUM upper main line
End lifter MDUL-E
Workstation MDUW
Lower Reverse Line
Main Line MDUM bottom reverse line

Fig. 15.14 A shop-floor system composed of a distributed control system exposing their production control and management functionalities as services

The final result of this engineering phase allows to see the same original service-oriented shop-floor depicted in Figure15.11 as presented in Figure 15.14. That is, a complex flat distributed automation system composed of a set of smart control devices shown in Figure 15.13 wrapping the mechatronics systems with their service view of Figure 15.12.

In order to show the gain in flexibility using SOA, the application was enhanced by performing a change in the production order from the ERP system, directly on the shop-floor (see Figure 15.15).

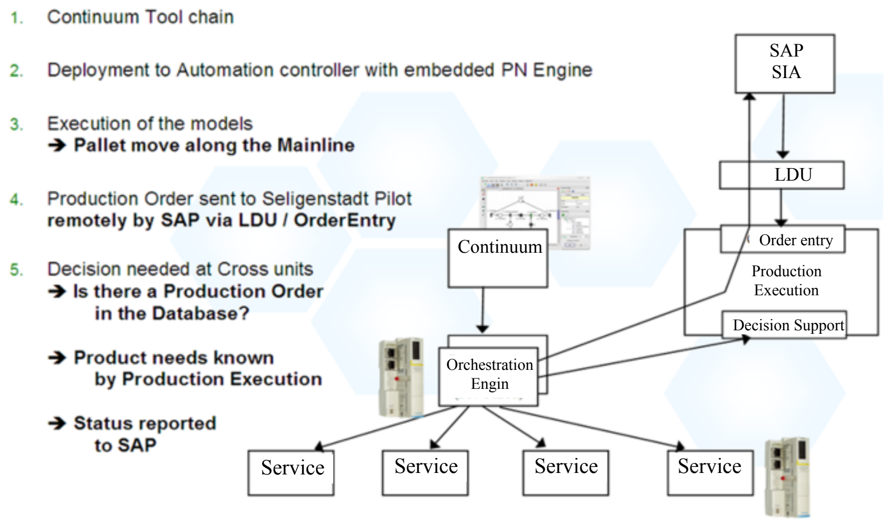

1. Continuum Tool chain
2. Deployment to Automation controller with embedded PN Engine
3. Execution of the models
➔ Pallet move along the Mainline
4. Production Order sent to Seligenstadt Pilot
remotely by SAP via LDU / OrderEntry
5. Decision needed at Cross units
➔ Is there a Production Order in the Database?
➔ Product needs known by Production Execution
➔ Status reported to SAP
SAP SIA
LDU
Order entry
Production Execution
Decision Support
Continuum
Orchestration Engin
Service
Service
Service
Service

Fig. 15.15 Production execution based on collaboration with enterprise systems and local services

Additionally only minimal assumptions about the concrete production line are present in the whole system design. The detailed production steps are stored in the production execution system, which is integrated in between of the Petri-net-based decision support module related to the smart devices and the ERP. The production order system is dynamically discovered and registers new orders in the ERP using the local discovery unit (LDU) of the SOCRADES integration architecture (SIA), further details of which are provided in Spiess *et al.* (2009).

Finally, it was possible to demonstrate the flexible integration and collaboration based on the idea of having a multi-site service-oriented enterprise in which the assembly of electromechanical components is performed in two geographically distributed assembly systems (both of them are similar to the system depicted in Figure 15.11) and production orders could be allocated to different sites. This allocation/re-allocation is done as an evaluation result of the best production facility available at the moment when a production request is made, or when – due to external factors – the production should be shifted to a different location.

The idea, as shown in Figure 15.16, is that similar production facilities are available in remote locations (e.g. Schneider Electric in Germany or Tampere in Finland). The prototypes developed and hosted in Tampere (TUT) and Seligenstadt (Schneider Electric) represent two different companies that are linked with business relations. These companies are "interconnected" via the Enterprise Applications that are hosted in Walldorf (SAP).

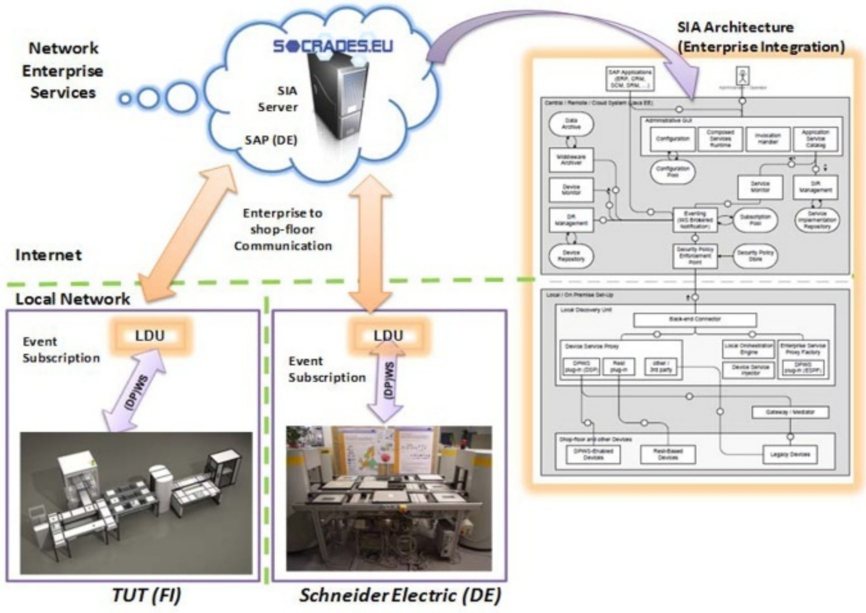


Fig. 15.16 Cross-company collaborative production based on the SOCRADES integration architecture

Both facilities provide electromechanical assembly capabilities using SOCRADES architecture; this means that the components of the production systems in these locations are abstracted and perceived externally as WS. At local level, each one of the facilities acts independently and can coordinate its service-enabled production system by using the SOCRADES tools. At global level, both facilities connect to a service-enabled ERP module provided by SAP, which is used for coordinating the production in the remote locations.

A network application (named LDU) is downloaded and this immediately provides discovery of devices and services (via the device profile for web services-DPWS) on the local network and connection to the backend system (the SIA). Different versions of the LDU can add-up functionalities, e.g., proxy also specific enterprise services at the local shop-floor, where they can be discovered and used by the devices and other services. LDUs provide a means for connecting and managing devices from different premises, without needing virtual private network connections to SAP premises.

The LDUs can discover local services and can interact directly with production execution systems exposed as WS. This is a typical example of hosted functionality on a network server at the provider side, where business services are being implemented/updated and the remote sites (Tampere/Seligenstadt) can interact over the network, with only minimal installations at their side (in order to interact with the business services). In this specific case, the software that interconnects each site is downloaded on the fly over the network.

The different sites involved are collaborating between them via interactions that previously were not possible or would require significant implementation efforts. As can be seen, this is an event-based approach where all sites are notified about the necessary status of the production in the other side, and where the enterprise systems have full visibility on the production and can re-arrange orders in order to meet business goals.

15.7 Conclusions and Future Work

The future factory should be seen as a system composed *of modular, dynamic reconfigurable and service-oriented collaborative systems*. Complex and dynamic mechatronics, control, communication and IT systems interact with each other, in a service-oriented manner, in order to achieve the goals at system-wide but also at local level (Kennedy *et al.*, 2008; Colombo and Karnouskos, 2009). To realize this, timely monitoring and control as well as open communication and collaboration in a cross-layer fashion are key issues.

Modern approaches such as the SOA paradigm when applied holistically can lead to the desired result. By considering the set of intelligent system units as a conglomerate of distributed, autonomous, intelligent, pro-active, fault-tolerant and reusable units, which operate as a set of cooperating entities, a new dynamic infrastructure, a system of systems, that is able to provide a better insight to its components to the higher levels and better react to dynamic business changes can be realized.

This work presented an overview of the engineering approach, methods and tools that have been specified and developed within the European Research and Development project SOCRADES (www.socrades.eu) and shows the results of the first set of successful applications in the area of elechtromechanical assembly systems, extending the concepts to geographically distributed service-oriented production sites.

Among many scientific and technological outlooks, an intensive study should be performed in spreading the application of the service-oriented paradigm in production control and management. This is having in mind that the roles of control and automation technology developers, producers of production components and IT systems is strongly influenced when all their products and solutions are to be integrated into a system of systems viewed/wrapped as services. A possible view could be that depicted in Figure 15.17.

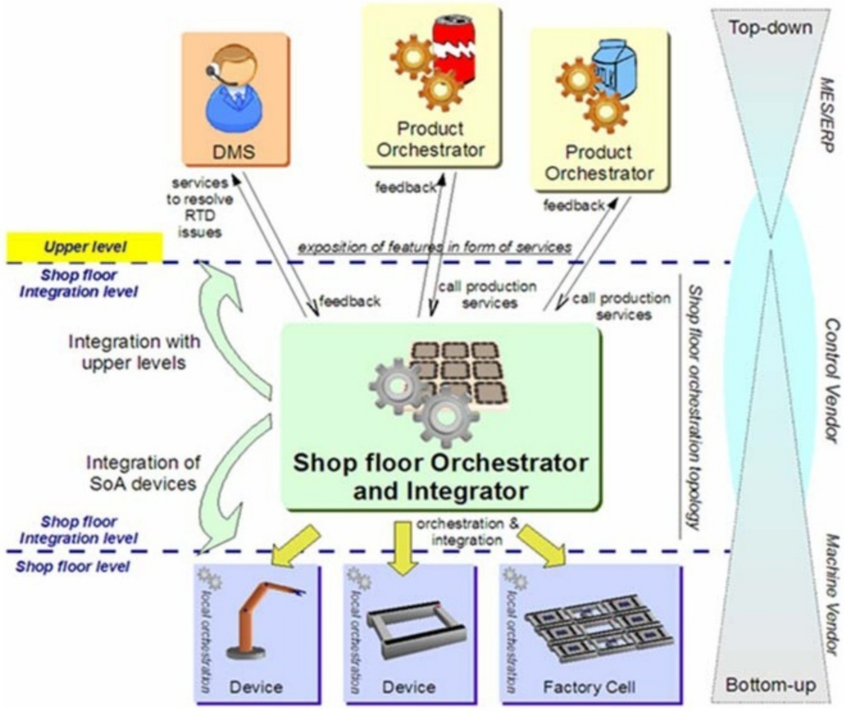

DMS
Product Orchestrator
Product Orchestrator
Top-down
MES/ERP
services to resolve RTD issues
feedback
feedback
Upper level
exposition of features in form of services
Shop floor Integration level
feedback
call production services
call production services
Integration with upper levels
Shop floor orchestration topology
Control Vendor
Integration of SoA devices
Shop floor Orchestrator and Integrator
Shop floor Integration level
orchestration & integration
Shop floor level
Machine Vendor
local orchestration
Device
Device
Factory Cell
Bottom-up

Fig. 15.17 Cross-layer systems integration facilitated by the SOA approach

Having a production system built according to the SOA paradigm, it can be seen as a conglomerate of distributed smart devices and systems able to interact at shop-floor level (horizontal) and in a cross-layer fashion integrating IT systems of the other levels of the enterprise architecture, a clear next step in a technology roadmap and research and development agenda is the realization of supervisory control, automation and production management functions as services.

Acknowledgements The authors would like to thank the European Commission and the partners of the European FP6 project "Service-Oriented Cross-layer infRAstructure for Distributed smart Embedded devices" (SOCRADES - www.socrades.eu) and the European FP7 project "Co-operating Objects Network of Excellence" (CONET – www.cooperating-objects.eu), for their support.

References

Cândido G, Barata J, Colombo AW. et al. (2009) SOA in reconfigurable supply chains: a research roadmap. Eng. Appl. Artif. Intell., 22(6):939–949

Colombo AW, Harrison R (2008) Modular and collaborative automation: achieving manufacturing flexibility and reconfigurability. Int. J. Manuf. Tech. Magmt., 14(3/4):249–265

Colombo AW, S. Karnouskos (2009) Towards the factory of the future: A service-oriented cross-layer infrastructure. In: ICT Shaping The World - A Scientific View, chapter 6, John Wiley and Sons, UK

Du R, Elbestavi, Wu S (1995) Automated monitoring of manufacturing processes, Part 1: monitoring methods. J. Eng. Ind., 117:121–130

Feldmann K, Colombo AW (1998) Material flow and control sequence specification of flexible production systems using coloured petri nets. Int. J. Adv. Manuf. Tech., 14(10):760–774

Fleisch E, Mattern F (2005) Das internet der dinge: ubiquitous computing und RFID in der praxis: Visionen, technologien, anwendungen, Handlungsanleitungen, Springer

Gorbach G, Nick R (2002) Collaborative manufacturing management strategies. White paper, ARC Advisory Group

Jammes F, Smit H. (2005) Service-oriented paradigms in industrial automation. IEEE Trans. on Ind. Info., 1(1):62–70

Jamshidi M (2008) Systems of systems engineering: principles and applications. CRC Press

Karnouskos S, Baecker O, de Souza LMS et al. (2007) Integration of soa-ready networked embedded devices in enterprise systems via a cross-layered web service infrastructure. In: Proceedings of the IEEE Conference on Emerging Technologies and Factory Automation (ETFA), pp. 293–300

Karnouskos S, Bangemann T, Diedrich C (2009) Integration of legacy devices in the future SOA-based factory. In: Proceedings of the 13th IFAC Symposium on Information Control Problems in Manufacturing (INCOM'2009), Moscow, Russia

Kennedy P, Bapat V, Kurchina P (2008) In pursuit of the perfect plant. Evolved technologist, ISBN 978-0978921866

Kreyszig E (1989) Introductory functional analysis with applications. Wiley and Sons

Leitão P, Mendes JM, Colombo WC (2008) Decision support system in a service-oriented control architecture for industrial automation. In: Proceedings of the 13th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'08), pp. 1228-1235, Hamburg, Germany, 15–18 Sep

Marron PJ, Karnouskos S, Minder D et al. (2009) Research roadmap on cooperating objects. European communities, ISBN 978-92-79-12046-6

Memmi G, Roucairol G (1979) Linear algebra in net theory. Lect. Notes in Comp. Sci., 84:213–223

Mendes JM, Bepperling A, Pinto J et al. (2009) Software methodologies for the engineering of service-oriented industrial automation: The continuum project. In: Proceedings of the 33rd Annual IEEE Computer Software and Applications Conf. (COMPSAC'09), Seattle, USA. 20-24 July

Murata T (1989) Petri nets: properties, analysis and applications. Proc. of the IEEE, 77(4):541–580

Namur (2007) http://www.namur.de/fileadmin/media/Pressespiegel/atp/atp_05_2007_DIN_EN_62264.pdf

Nick R, Polsonetti C (2003) Collaborative automation: The platform for operational excellence. White paper, ARC Advisory Group

PERA (2006) Purdue reference architecture http://pera.net, http://iies.www.ecn.purdue.edu/IIES/PLAIC/

Pfadenhauer K, Kittl B, Dustdar S et al. (2006) Shop-floor information management and SOA. Lect. Notes in Comp. Sci., 4103:237–248

Silva M, Valette R (1989) Petri nets and flexible manufacturing. Advances in Petri nets. Lect. Not. In Comp. Sci., 424:374–417

Spiess P, Karnouskos S, Guinard D et al. (2009) SOA-based integration of the Internet of things in enterprise services. In: Proceedings of the IEEE International Conference on Web Services (ICWS09), Los Angeles, CA, USA, 6–10 July

Chapter 16
A Service-oriented Shop Floor to Support Collaboration in Manufacturing Networks

J. Barata, L. Ribeiro[1] and A.-W. Colombo[2]

Abstract The present work addresses the problem of shop-floor agility, presenting it as the fundamental cornerstone for true agility and responsiveness of an enterprise willing to participate in highly dynamic collaborative organizations and supply chains. Clearly, as the economic climate toughens, the exploration of the increasingly volatile business opportunities requires such complex organizations. The feasibility of the architecture proposed is demonstrated in a pilot implementation in a near-real shop-floor. Emerging web standards such as the device profile for web services were used to guarantee cross-layer/abstraction interoperability ensuring that the shop-floor reacts positively to adjustments in the supply chain.

Keywords Service-oriented shop-floor, collaboration, manufacturing networks

16.1 Introduction

One of the most fundamental issues in current manufacturing companies in order to address the turbulent economic environment they are currently facing is agility or responsiveness. A responsive or agile company is one that is able to respond swiftly and smoothly to fast internal and external changing conditions. It is fundamental to clarify that agility or responsiveness are wider concepts than flexibility since it means that the company adapts to changing conditions within an acceptable time frame.

Being responsive and adaptable means that a company is prepared to participate in dynamic supply chains or manufacturing networks which is an important requisite in today's manufacturing environments. In fact anyone committed to surviving in the current economic environment must be able to enrol in different

[1] J. Barata (✉), L. Ribeiro
Universidade Nova de Lisboa /UNINOVA, Quinta da Torre, 2829-516 Caparica, Portugal
e-mail: jab@uninova.pt

[2] A.-W. Colombo
Schneider Electric GmbH, P&T/H6O HUB, Steinheimer Str 117, 63500 Seligenstadt, Germany

supply chains. The issue is that in turbulent times supply chains are becoming more and more dynamic. Cooperation occurs with different companies simultaneously and within different time frames.

Supporting manufacturing networks is therefore a key aspect in order to have responsive and adaptable companies. The research question is then what methods and tools are necessary to develop for responsive and adaptable companies. In the work described within this chapter the authors' main argument is that this can only be possible if responsiveness and agility are achieved at lower levels.

It is therefore not surprising that research effort is being directed towards making the shop floor available to business tools as they increasingly require run-time shop-floor information to be used in decision-taking concerning rearrangements in the supply chain.

This is being supported by recent developments in information technologies (IT) and the increasing computing power available in embedded devices that allows a pervasive and unprecedented use of artificial intelligence. In the industrial control community research on the next generation of production systems is undergoing.

The service oriented infrastructure later detailed follows this rationale and fulfils the high reconfiguration requirements that enable agility in the shop floor and consequently in the supply chain by exploiting local intelligence. Furthermore the openness of the architecture allows external systems and tools to collect relevant data from the devices in the shop floor. The proposed infrastructure uses the device profile for web services (DPWS) stack developed under the SIRENA project (SIRENA 2006) as it targets devices with reduced computing power, and applies it to low-granularity components such as grippers, robots, conveyors, etc., to enable intelligence and communication. The systems being addressed are no longer static immutable units. Instead they are dynamic compositions of intelligent modules that interact to execute the tasks they were designed for (assemble) and also to support various activities in their life cycle. Based on these intelligent building blocks, shop floors become more autonomous, responsive and adaptable.

This chapter starts by addressing what agility looks like in the context of manufacturing and how it affects the shop-floor control requirements. After, the concept of collaborative network is discussed in order to provide the reader with insight about this new dynamic organizational form of manufacturing companies and how this dynamic structure calls for highly dynamic and changeable (agile) shop-floor controllers. After these two first sections (16.2 and 16.3), the chapter starts discussing the service-oriented architecture based on DPWS developed to support agility and responsiveness at lower levels (control). First, the service-oriented architecture (SOA) method to support agility and collaboration is discussed in Sect. 16.4, and then in Sect. 16.5 the SOA-based architecture to support agility is presented. To show that the proposed approach is not just a theoretic idea without any proof of concept, an implementation in which agility and responsiveness was achieved is described in Sect. 16.6. Finally, Sect. 16.7 discusses the main conclusions of the chapter.

16.2 Agility in Manufacturing

Changing and uncertain markets, society, or technology are so omnipresent that a paradigm that can handle turbulence is demanded. This manufacturing paradigm is called agile manufacturing and the term was coined by Nagel and Dove in the report "21st Century Manufacturing Enterprise Strategy" they developed at the Iacocca Institute (Nagel and Dove, 1992).

Agile manufacturing is a step forward with respect to anthropocentric and lean manufacturing because those paradigms can only satisfy a controlled environment, while agile manufacturing deals better with things that cannot be controlled (Maskell, 2001). Agility is the ability to thrive and prosper in an environment of constant and unpredictable change (Goldman *et al.*, 1995), and is required not only to accommodate change but also uncertainty. Agility is more than being flexible. The flexible manufacturing systems already developed in the 1980s cannot cope with uncertainty. Even if they can produce a variety of products and accommodate some changes on demand, they can only do it if these variations are predictable. The goal of manufacturing systems is now to provide high quality products instantaneously in response to demand (Davidow and Malone, 1993).

If the requirements imposed for agile manufacturing had been needed before the advent and dissemination of information and communication technologies (ICT), it would not have been possible to implement it effectively, because ICT is one key technology enabler. The successful implementation of agile manufacturing requires the following points (adapted from Vernadat, 1999; Barata and Camarinha-Matos, 2000; Barata *et al.*, 2001; Camarinha-Matos and Barata, 2001; Hormozi, 2001; Leitão *et al.*, 2001; Maskell, 2001; Onori, 2002):

- **Political decisions** – it is important to create, for instance, regulations to help cooperation and innovation.
- **Business cooperation** – with the increase of short-term relationships between suppliers and customers, it is natural that companies should be able to diversify cooperative relationships as much as possible. The need for agile business structures to facilitate the creation of dynamic networks of enterprises is mandatory. The ability to create virtual organizations (Vos), which are opportunistic alliances of core competencies across several firms to provide focused services and products, is thus the limit in terms of cooperation.
- **Customer focus** – although this is related to the previous point it was decided to separate it to stress how important it is to create a philosophy to focus the company on the customer. Solutions which involve product and services must be sold to customers to increase the product value and the awareness of the customer to pay more for that product. The addition of value to the product may even be obtained by cooperation with other companies, which is achieved by the previous point.
- **Information technology** – the requirements needed to keep a close relationship with customers and suppliers can only be effective with good computa-

tional support. Production systems need also computational support to create flexible and agile shop floors. Shop-floor equipment such as robots, computed numerical control, and transporters, all require computers. Finally ICT is also fundamental to support the fast design of products. Concurrent systems to involve the customer, suppliers, and the manufacturing people in the product life cycle can only be achieved using computers.

- **Processes reengineering of the company** – this involves not only identifying what processes should exist but also redesigning them. Reorganization must be a routine. In addition, more than one organizational structure might be needed at the same time.
- **New work organization** – The tayloristic work organization of division of labour has to be replaced by an organization based on teams and cooperation. On the other hand, workers need to be more skilled because the company is more complex and they can intervene more and have more autonomy. The workers need to be highly educated and trained. The anthropocentric work organization approach seems to be well indicated for agile manufacturing.
- **New agile shop-floor paradigms** – the current approaches used on the shop-floor like traditional flexible manufacturing/access systems are not well indicated to deal with turbulence. One important limitation of those strategies is their inability to deal with the evolution of the production system life cycle. A methodology that could integrate the various aspects related to the life cycle of the production systems is thus very important for an agile shop floor. This aspect is much wider than the problem of flexibility and the addition of computer equipment. It involves a careful study of the actors, the processes, and the areas that are involved in the production system and then the development of a methodology that helps the integration of these areas and supports their life cycle evolution.
- **Willingness to change** – this aspect must be present in all the previous points. The agile manufacturing can only be successful if all its actors are constantly monitoring the situation around them and willing and prepared to react whenever it is necessary. This might be achieved by a system of incentives. The company needs to have a clearly defined vision of where the company is going to, and how these objectives will be met. Being responsive is the key word for success. This applies to supportive research and development and academia as well.

In the context of this chapter the most important aspect to keep in mind is that agility implies ability to deal with very frequent and unpredictable changes, which must be handled within a very short time. These changes imply that shop-floor control must also be agile in the sense that it supports changes handled in a very short time. By having an agile shop-floor companies can easily reconfigure and adapt their network ties and are therefore able to create agile supply chains.

16.3 Collaborative Networks

Although the concept of collaborative-networked organization is sometimes confused with the concept of VOs, they are not the same because the latter is just one type of the former. Collaborative-networked organizations are not circumscribed to the business domain.

Definition 16.1 *A collaborative-networked organization is any group of autonomous entities, which may be organizations, people, or artificial agents that have together formed a cooperative dynamic network to reach individual or group benefits.*

With this definition different types of collaborative organization can be considered. Camarinha-Matos (2002) considered the following forms:

1. **VOs** – It includes both business and non-business organizations. Example of the latter is the formation of humanitarian relief operations, military operations, and governmental and non-governmental organizations.
2. **Professional virtual communities (PVCs)** – These are organizations of people that share the same professional interests. The members are bound to some social rules resulting from the commitment of their members to the PVC that constrain their behaviour. From within PVCs virtual teams can emerge to deal with subjects relevant to the organization, for example, ethical issues.
3. **E-science** – These are communities of scientists that use virtual labs to support their work.
4. **Advanced supply chains** – new advanced supply chains are composed of companies whose relationships are more equalitarian than based on the predominance of one over the others, as was the tradition.

Considering that the life cycle of a VO is composed of VO planning and creation, VO operation, and VO dissolution the main difficulties associated with each phase are (Camarinha-Matos, 2002):

- **VO planning and creation** – some of the obstacles include the lack of appropriate support tools, namely for: partners search and selection (e-procurement), VO contract bidding and negotiation, competencies and resources management, task allocation.
- **VO operation** – the main challenges here are: well-established distributed business process management practices, monitoring and coordination of task execution according to contracts, performance assessment, inter-operation and information integration protocols and mechanisms, etc. Further problems include the lack of common anthologies among the cooperating organizations, derivation of the information visibility regulations based on contracts, the proper support for socio-organizational aspects, e.g. lack of a culture of cooperation, the time required for trust building processes, the need for BP reengineering and training of people, etc.

- **VO dissolution** – this phase is practically not covered by research projects.

Since our service-oriented approach to support collaboration or agile supply chains is itself inspired by collaborative networked organizations, particularly VO, the problems highlighted above are also expected to be found when forming coalitions or consortia of manufacturing components (SOA devices). Undoubtedly our coalitions need to be planned and created, operated, and dissolved. The following research topics from the VO world have influenced our approach:

- Selection of partners with the right skills to answer the requirements needed by the VO.
- The coordination mechanisms used by VO members.
- Value system. This includes the "quality" of each company and the quantification of their performance within the VO.
- Contracts to regulate behaviour.
- Mechanisms to enable and regulate the dynamic interaction that is needed by VO participants. This includes the aspects that facilitate the interconnection between different entities in a fast way (methods and tools for interoperability).

The fundamental aspect of collaborative networks is that individual nodes (companies) must be able to quickly adapt and reconfigure whenever they need to change from one group structure (coalition) to another. The only possibility for a company to be able to do this is if its shop-floor control is easily reconfigurable and changeable in a very short time.

16.4 Service-oriented Architecture as a Method to Support Agility and Collaboration

Successful collaboration in manufacturing networks requires the use of an abstraction/metaphor that renders heterogeneous systems seamlessly interoperable and modular. Only after this has been achieved can the truley potential for collaborative interaction be explored. In this context, it is fundamental to support the following features:

1. **Modularity** – the platform should rely on distributed modules (building blocks) that are combined to deliver certain functionality.
2. **Autonomy** – the modules are created independently of each other and provide self-contained functionalities, working as black boxes for the remaining system.
3. **Proactivity** – the modules should be able to follow their own agenda and decide autonomously under the limits of their purpose and design.
4. **Interaction** – the modules must interact according to certain rules or agreements to deliver a functionality that is a combination of individual contribu-

tions. In some paradigms the principle that the whole exhibits a functionality that is bigger than the sum of the parts is underlying.
5. **Structure** – all the modules play a certain role in the system. The definition of a structure supports the progressive encapsulation of complexity.
6. **Hierarchy** – some modules may be allowed to overrule or reinforce decisions/actions taken by other modules.
7. **Dynamics** – the systems addressed are not static entities. Most of the approaches support the idea of evolution, either by introducing changes in the environment or learning.
8. **Heterogeneity** – the systems targeted by these paradigms are heterogeneous requiring a careful interface definition between entities in the environment.
9. **Performance** – critical decisions should have real-time support.

Can an SOA tackle or address these points and therefore contribute to the effective implementation of collaborative manufacturing network? The preferred mechanism for SOA modelling is the web service. The Web Services Working Group of the World Wide Web Consortium defines a web service as "a software system designed to support interoperable machine-to-machine interaction over a network (see Booth *et al.*, 2004). It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web Service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards" (see Box *et al.*, 2000). Table 16.1 attempts to show how well web services meet the requirements stated earlier.

Table 16.1 Web service vs. collaborative networks requirements

Are web services...	Argument
Modular?	There are no direct dependencies between the web services as they are structurally decoupled.
Autonomous?	Web services are created independently of each other. Dynamic composition/orchestration may bind periodically web services without changing, however, its primitive function.
Proactive?	Typically the focus is on the interface definition rather than on the implementation. There are no technical constraints regarding proactivity it is normally a design decision.
Interactive?	Modularity supports the idea of reusing self-contained functionality according to the application requirements. Composition necessarily implies interaction. The self-describing interface specifies the interaction rules and mechanisms.
Structural and hierarchic?	The self-describing nature of the interface allows the implementation of services operating in distinct abstraction levels.
Dynamic?	There are still significant research challenges regarding the implementation of truly dynamic SOA-based ecosystems.

Table 16.1 (continued)

Are web services...	Argument
Able to support heterogeneous systems?	The use of web standards ensures cross-platform interoperability and the self-describing format supports to some extent harmonization from an information processing point of view.
Real-time?	Greatly depends on the stack implementing the web services. Minimal stacks such as DPWS are able to perform in some real-time scenarios.

Although a significant share of the research in SOA focuses on modelling and supporting inter-enterprise relationships, there is a favourable convergence of factors that are rendering it attractive in the establishment of automated networks of devices, namely: the availability of affordable and high-performance embedded devices, the expansion and low cost of Ethernet-based networks and their acceptance in the industrial domain, the ubiquitous nature of the internet, the existence of lightweight, platform-agnostic communication infrastructures, etc.

These factors have motivated recent research projects on the application of SOA for automation:

- **SIRENA (SIRENA 2006)** – award winning project that targeted the development of a service infrastructure for real-time embedded networked applications (Jammes and Smit, 2005). An implementation of a DPWS stack has been produced under the framework of this project and successfully applied in device level automation test cases (Jammes *et al.*, 2005, 2007).
- **SODA (SODA 2006 - 2008)** – creation of a service-oriented ecosystem based on the DPWS framework developed under the SIRENA project. SODA focus is on exploring the potential of DPWS advance orchestration and composition capabilities.
- **SOCRADES (SOCRADES 2006)** – development of DPWS-based SOA for the next generation of industrial applications and systems.
- **InLife (InLife 2006)** – development of a platform for integrated ambient tntelligence and knowledge-based services for optimal life-cycle impact of complex manufacturing assembly lines. The DPWS toolkit resulting from SIRENA was applied to a pilot assembly cell as its basic IT infrastructure support. Advanced functionalities including self-monitoring/diagnosis/reporting were implemented at device and process levels (Barata *et al.*, 2007ab; Ribeiro, 2007).

Despite the advances in the applications of SOA (and as stated in Table 16.1) there are still open challenges as pointed out in a roadmap presented by Papazoglou *et al.* (2005) that identifies the following research areas: service foundations, service composition, service management, service design and development. More specifically: dynamically reconfigurable run-time architectures, services discovery, autonomic composition of services and orchestration, self-* (self-configuring/healing/diagnosing,…) services, design principles for engineering service applications. While some of these challenges can be eased by emergent standards such as BPEL4WS (Andrews *et al.*, 2003) and WSCI (Arkin *et al.*, 2002) others, specifically service composition in dynamically reconfigurable run-time architectures, are harder to tackle in heterogeneous systems, which are typically SOA's target environments.

Additionally, as stated in (Huhns and Singh (2005) most web services specifications do not properly support transactions which constitutes a significant implementation barrier in a wide range of systems.

Ribeiro *et al.* (2008)a considered the benefits of borrowing multi-agent (MAS) principles and introducing them in SOA taking service oriented modelling as a starting point. A side-by-side analysis is presented in Table 16.2.

Table 16.2 Comparative analysis between SOA and MAS (Ribeiro *et al.*, 2008a)

Characteristics	SOA	MAS
Basic unit	Service	Agent
Autonomy	Both entities denote autonomy as the functionality provided is self-contained	
Behaviour description	In SOA the focus is on detailing the public interface rather than describing execution details	There are well-established methods to describe the behaviour of an agent
Social ability	Social ability is not defined for SOA nevertheless the use of a service implies the acceptance of the rules defined in the interface description	The agents denote social ability regulated by internal or environmental rules
Complexity encapsulation	Again, the self-contained nature of the functionalities provided allows hiding the details. In SOA this encapsulation is explicit	
Communication infrastructure	SOA are supported by web-related technologies and can seamlessly run on the internet	Most implementations are optimized for LAN use
Support for dynamically reconfigurable run-time architectures	Reconfiguration often requires reprogramming	The adaptable nature of agents makes them reactive to changes in the environment.
Interoperability	Assured by the use of general purpose web technologies	Heavily dependent on compliance with FIPA-like standards
Computational requirements	Lightweight implementations like DPWS (Chan *et al.*, 2006) guarantee high performance without interoperability constraints	Most implementations have heavy computational requirements

This argumentation has led the authors to the implementation of a generic communication interface for web services (Ribeiro *et al.*, 2008b). While the work does not attempt to solve all the emerging challenges in the area it addresses dynamic composition and orchestration in devices with limited computing power. The rationale was the use of a common service description that any service could consume to request the execution of task from any other service. Although in principle this approach applies to any interaction pattern, a FIPA request protocol (FIPA, 2002) like interaction was considered as it specifically targets peer-to-peer interaction. When launched on the manufacturing network each device broadcast its functionalities. Existing devices then store that information in tuples with the following form:

Service_Specs(endpoint, device, skill, neighbourhood_list)

The "endpoint" is the address of the web service where the functionality or "skill" is being hosted by a given "device". The "neighbourhood_list" value specifies which devices or entities can safely interact with the announced skill. All the entities behave as generic skill-processing machines as specified in Ribeiro *et al.* (2008ab). The code explosion phenomena and the additional complexity introduced by necessity of handling heterogeneous services description are controlled (Figure 16.1).

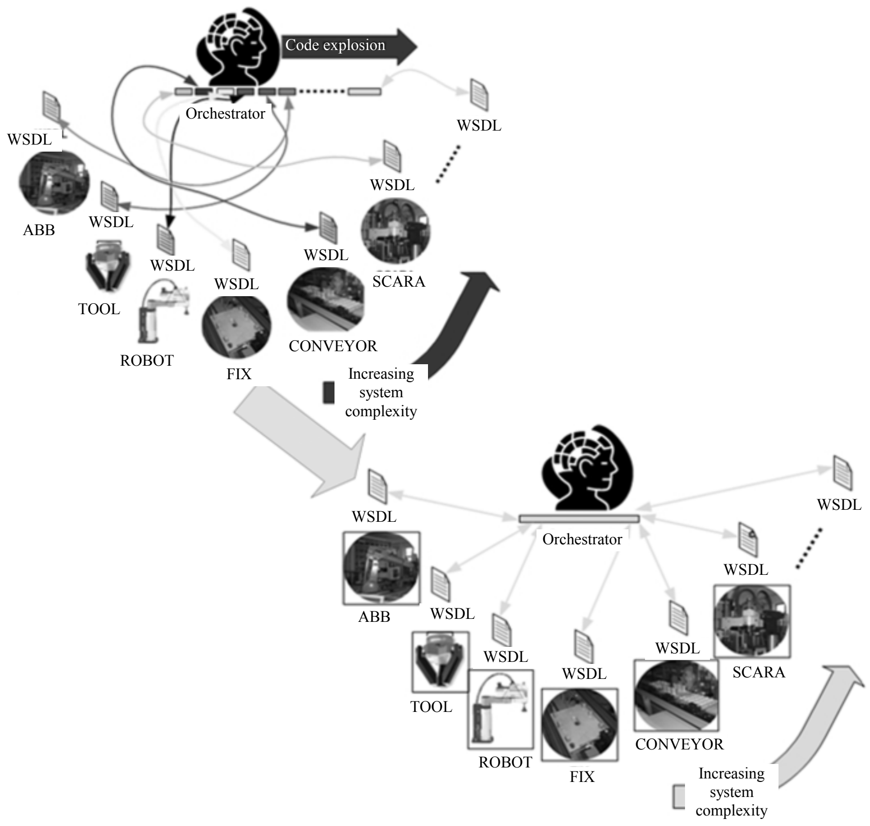


Fig. 16.1 Reduction in the overall information process complexity

As shown in Figure 6.1, the generic communication interface restricts the number of message handlers that have to be implemented therefore supporting run-time changes in the participants in the system.

In the next section, the architecture for collaborative manufacturing based on these principles will be presented.

16.5 Architecture

16.5.1 The Role of Componentization

The successful implementation of collaborative automation requires firstly a paradigm shift.

Centralized control approaches tend to perform rather well whenever process optimization is required. This is achieved at the cost of tightly coupling processes and the devices associated with them. When the focus is on adaptation, robustness and responsiveness these approaches perform poorly, mainly due to lack of redundancy in the decision process.

If rather than implementing a process as the coordination of input/output signals one implements it as an explicit collaborative set if interaction (Figure 16.2) where each device plays a role and supports a function the required shift to take full advantage SOA is made.

In fact, in an assembly process, several participants can be identified: grippers, conveyors, routing devices, fixing devices, warehousing devices, manipulators, pallets, etc.

Giving autonomy and decision capabilities at local (device/process) level, under the design and purpose scope of the device, introduces the required margin to make the shop floor more agile. This agility can then be explored either during configuration/reconfiguration of the system or in run-time to influence the behaviour of the shop floor to meet new production requirements or circumvent faults.

From the user's point of view the system, rather than a collection of hardwired devices, becomes as decoupled set of service providers whose functionalities can be (re)combined to generate new value. The focus is clearly on reconfiguration rather that the traditional reprogramming as shown in Figure 16.2.

In the present architecture, three generic components are considered:

1. Coalition leader service (CLS) – entity in charge of aggregating and orchestrating other services. Behaves as a generic process executor.
2. Manufacturing device service (MDS) – abstracts each individual manufacturing component: tools, robots, conveyors, etc.
3. Service to machine interface (SMI) – the SMI harmonizes legacy equipment with the IT platform, the wrapper approach is used.

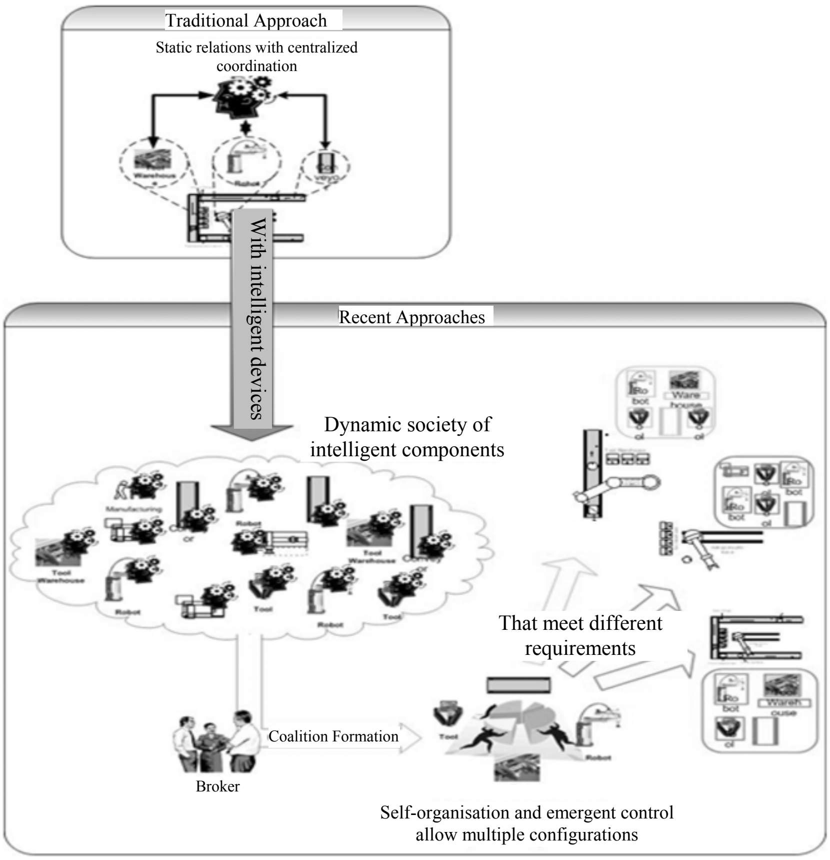

Traditional Approach
Static relations with centralized coordination
Warehouse
Robot
With intelligent devices
Recent Approaches
Dynamic society of intelligent components
Manufacturing
Robot
Tool Warehouse
Robot
Tool
That meet different requirements
Coalition Formation
Broker
Tool
Robot
Self-organisation and emergent control allow multiple configurations

Fig. 16.2 Paradigm shift from static interaction to dynamic collaboration

Furthermore, each componentized device is generic for classes of equipment and stores relevant information about itself (documentation, life-cycle parameters, state, etc.), which is publicly available for the remaining system through its IT frontend. That information can be used for a wide range of activities. In the present context, the underlying principle was that each service should support a critical dimension of an assembly cell life-cycle maintenance specifically condition-based maintenance (CBM) (Iung *et al.*, 2007). The basic principle of CBM is the support of maintenance decisions through the continuous monitoring of devices' health status (Jardine *et al.*, 2006). Therefore, when componentizing the environment the main target is the establishment of an architecture that supports control and diagnosis in dynamic and evolvable environments where devices are often plugged and unplugged and yet allows maintenance and fault information to be processed and forwarded to the entities that could best handle it (including: maintenance personnel equipped with mobile devices, remote systems, other devices, etc.) as proposed in Ribeiro *et al.* (2009ab).

The rules for composing and aggregating the services as well as the behaviour of each building block (SMI, MDS and CLS) for control and monitoring/diagnosis purposes will now be detailed.

16.5.2 Service Exposure, Composition and Aggregation

Each intelligent service, regardless of its type, exposes a certain set of skills. A skill is defined as an action that a given device knows how to execute and therefore is able to offer to the remaining system. Furthermore, in this document there is a distinction between simple and complex skills. A simple skill is atomic while a complex skill is the result of composing skills regardless of their type to deliver new functionality.

Typically, devices that directly access the hardware are assigned an MDS service and expose simple skills. In this sense, the only tasks/skills a gripper knows how to perform are basic grip and ungrip. Similarly a manipulator is only aware of the move/move linear, change speed, etc., functionalities.

MDSs are composed so that through interaction the system is able to deliver the required functionality and are aggregated under coalitions. For the purpose of this chapter a coalition is defined as a temporary organized group of services that jointly accomplish a certain task in a cooperative manner.

Each MDS when appearing in the network registers itself under a given scope. Devices under the same scope belong to the same coalition.

From an architectural point of view the CLS is the coordinating centre of a coalition. When launched in a network the CLS will search for services under its scope. In this context, the CLS must be able to monitor and orchestrate other MDSs and/or CLSs to accomplish a sequence of actions that defined the desired functionality.

The devices participating in a coalition directly affect the number of complex skills a CLS can expose. Under these circumstances the CLS must react to changes in the coalition: joining devices may enable the use of more complex skills or reinforce existing ones, offering redundancy, while leaving devices work inversely. To a certain extent one can state that complex skills emerge. This emergence is nevertheless in accordance to the emergence rules and constraints defined in the CLS. This mechanism is conceptually depicted in Figure 16.3.

It is important to stress that when a device is set running for the first time in the system the CLSs must be aware of the possible emergent skills using that device. That is, the new complex skills' description must be communicated to the system.

When this happens the CLSs update their knowledge base with the new description and the system becomes plug-and-play for all the devices with the same set of skills.

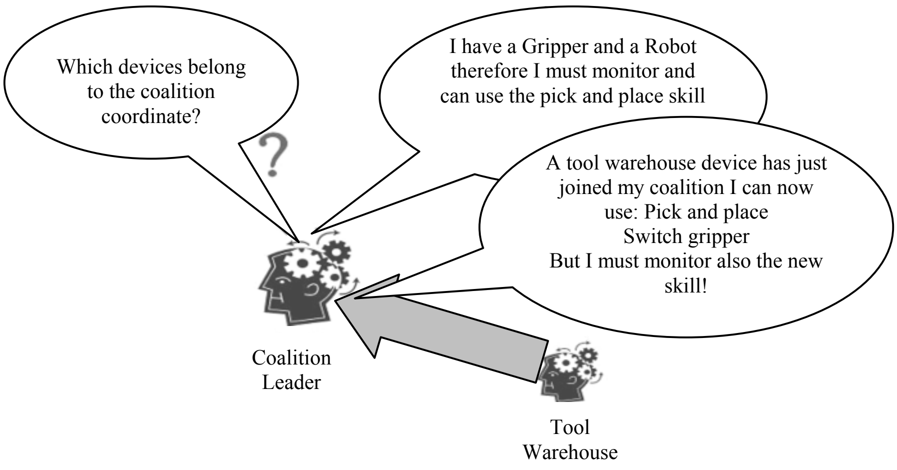

Which devices belong to the coalition coordinate?
I have a Gripper and a Robot therefore I must monitor and can use the pick and place skill
A tool warehouse device has just joined my coalition I can now use: Pick and place Switch gripper But I must monitor also the new skill!
Coalition Leader
Tool Warehouse

Fig. 16.3 Adapting dynamically to changes in the coalition

These features of the architecture improve overall scalability and plugability, as any combination of services is possible, and allow one to progressively encapsulate complexity through the systematic use of CLSs without constraining individual access to any service as shown in Figure 16.4.

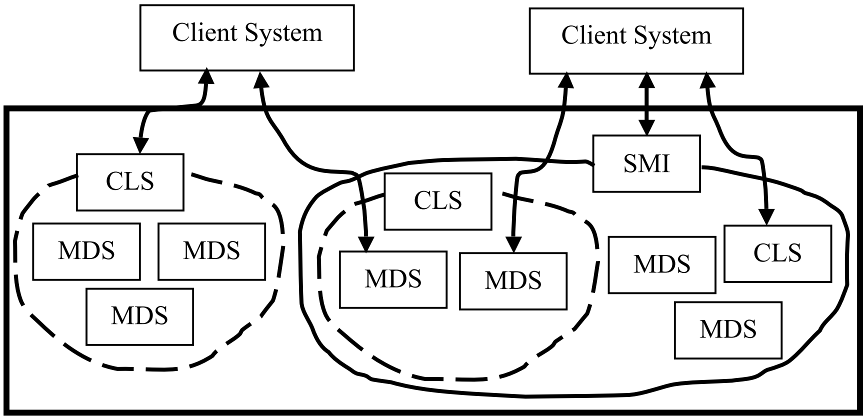


Fig. 16.4 Architecture scaling, multiple configurations of services are possible

16.5.3 The Role of Orchestration in Control, Monitoring and Diagnosis

As clearly depicted in Figure 16.4 the architecture comprises two fundamental dimensions: device (MDS) and system (CLS). Both explore the so-called self-* capabilities (Luck *et al.*, 2005) following the principle that: if all the participating entities exhibit self-diagnosis/healing, the system, as a whole, will exhibit similar properties (which are not necessarily the sum of the individual contributions). Unfortunately interaction faults and fault propagation are problems that often fall beyond individual capabilities and may require coordinated actions (Barata *et al.*, 2007ab).

To overcome these conditions the ability of a CLS to perform (re)orchestration is fundamental as it is responsible for controlling, monitoring and diagnosing other services and their interaction.

The present architecture takes advantage of the generic communication interface (earlier mentioned) to route messages around the system to the place where that data can be best handled. This mechanism will be defined in a bottom-up fashion taking as starting point the MDSs.

At MDS level each device senses the environment. Sensorial data is then fed to a generic reasoning mechanism. Depending on the MDS, the reasoning mechanism will consult the correspondent behavioural-logic-qualitative model of the device and, considering the implicit goal of explaining contradictions, will attempt to diagnose a fault. This reasoning mechanism is supported by the ACORDA engine (Lopes and Pereira, 2006).

An ACORDA agent can prospectively explore "what if" scenarios while formulating abdicative explanations for the observations taken. Distinguishing between the possible scenarios is done using a preference mechanism. Moreover, the agent is capable of triggering additional experiments to gather extra information "so more informed choices can be enacted, in particular by restricting and committing to some of the abdicative explanations along the way" (Lopes and Pereira, 2006).

In order to remove a contradiction it is necessary to assume at least one hypothesis of faulty behaviour. In this context the set of possible faults ranges from failed executions to abnormal sensor readings including unknown and unpredictable faults. The preference rules are then applied using the current knowledge of the system. Typically the base knowledge used in the model will be devised using structured domain data.

Although at MDS level the system will correctly identify and solve most of the faults, under some circumstances, a given fault may be masking another that is external to that service (a propagated fault or an interaction fault).

As stated earlier, interaction monitoring is coordinated by the CLS devices. What happens when an MDS fails a self-recovering action or misdiagnoses a fault?.

Under normal circumstances the CLS will not be notified of a fault from which an MDS recovered. Nevertheless, when a recovery action fails the MDS will report the fault to the CLS that is in charge of that coalition. This CLS, monitoring the execution of the process, will immediately stop executing that process and verify whether a specific recovery sequence was defined in its knowledge base to react to the event being reported.

If a specific action can be taken the CLS will re-orchestrate the coalition to execute it. It will either succeed, in which case the CLS will re-execute the failed action, or fail causing that CLS to forward the fault further so that it can be handled elsewhere. Successful recovery actions in a subsystem (CLS) are transparent to the remaining system.

This approach presents several advantages. CLS can be defined for monitoring/diagnostics purposes only regardless of control coalitions. Furthermore services can probe for monitoring/diagnostic information in a multilevel structure with different semantics. The same is to say that if a certain CLS is only interested in the completion of a pick-and-place operation it does not have to be informed of robot-related faults. In short, coalition configuration and CLS behaviour can be tuned to meet different requirements.

Orchestration also provides fundamental support in deciding whom to interact with, which information to request and the next course of action. As stated before, componentization allows seamless interaction between all participants in the process.

16.6 An Implementation

16.6.1 Manufacturing Device Service Implementation

MDSs are coupled to given hardware either using it directly or taking an SMI as intermediary. Although an MDS implements functions that are specific for a given class of assembly equipment the information flux inside the service is similar and is as depicted in Figure 16.5.

16.6.2 Coalition Leader Service Implementation

The coalition leader service is in charge of orchestrating other services under a given coalition. Furthermore the CLS (Figure 16.6) supports diagnosis at process level and has a crucial role in recovering from faults that MDSs alone cannot recover from.

All the CLSs use the same code supported by three modules: MessageManager, SkillExecutionManager and DiagnosisManager. Although a significant effort was put into developing generic code at CLS level, some sections of the code (representing either models or knowledge) still must be programmed or passed to the system in a suitable format.

In this sense, for the purpose of executing composed skills the CLS requires that the execution list of the composed skill is established. While for some operations this could easily be conceived (for instance in a pick-and-place operation all arguments are known *a priori*) for other operations the arguments of certain actions in the execution list require data that is created in run-time. In those cases an update function must be programmed to instruct the system on how to update the execution list.

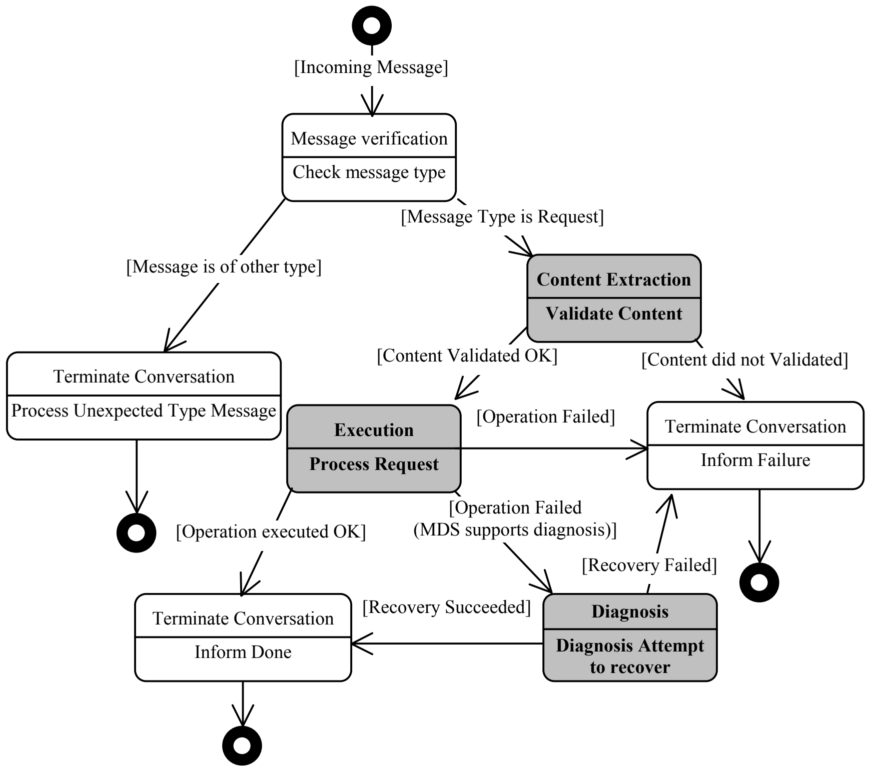


Fig. 16.5 Processing a request where coloured states are device specific

16.6.3 Application Example: a Collaborative Pick-and-place Operation

As a test case for the described architecture a pick-and-place operation has been considered. One robot MDS was instantiated and provided the MOVE skill. A gripper MDS was instantiated to enable the OPEN_GRIPPER and CLOSE_GRIPPER skills. A CLS was instantiated to manage the pick-and-place process. The final client of the system is a pallet containing the object subject to the operation. The pallet itself may be another CLS and may be coordinating other process that includes the current operation. The sequence of actions is the following:

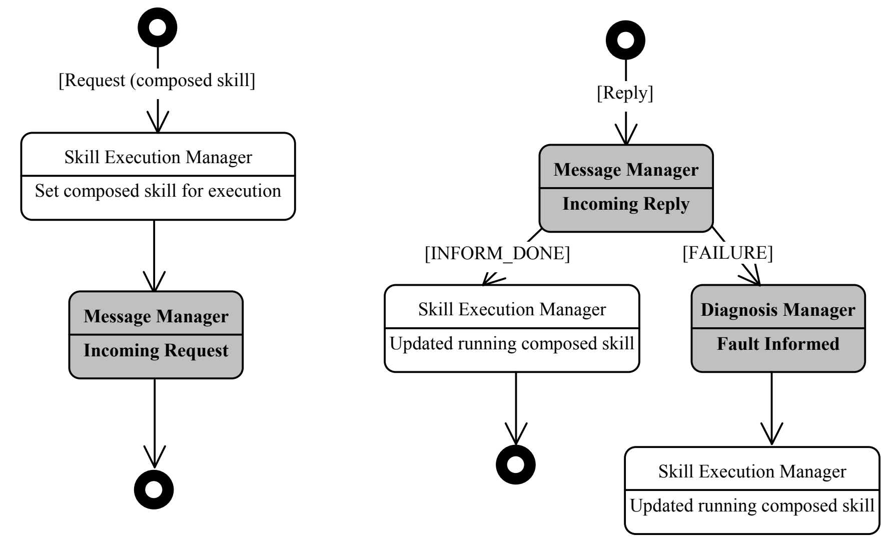


Fig. 16.6 Global functioning of a CLS device

1. Move to an approximation point of the pick position (robot MDS).
2. Open gripper (gripper MDS).
3. Move linearly to pick position (robot MDS).
4. Close gripper (gripper MDS).
5. Move linearly to safe point (robot MDS).
6. Move to an approximation point of the place position (robot MDS).
7. Move linearly to place position (robot MDS).
8. Open gripper (gripper MDS).
9. Move linearly to safe point (robot MDS).

The sequence is achieved by enabling dialogues between the CLS and the MDSs. If there is a failure in the process the participants involved in the fault will enter diagnosis mode and attempt to recover. The fault information is progressively propagated upwards when the entity in charge is unable to handle it. This process stops at a level where there is enough information to handle the fault. Recovery action may include finding replacement parts to perform the pick-and-place operation, roll back the pick-and-place to a given step, and forward the operation to another CLS with a similar pick-and-place skill. The collaborative process where the operation runs successfully is presented in Figure 16.7.

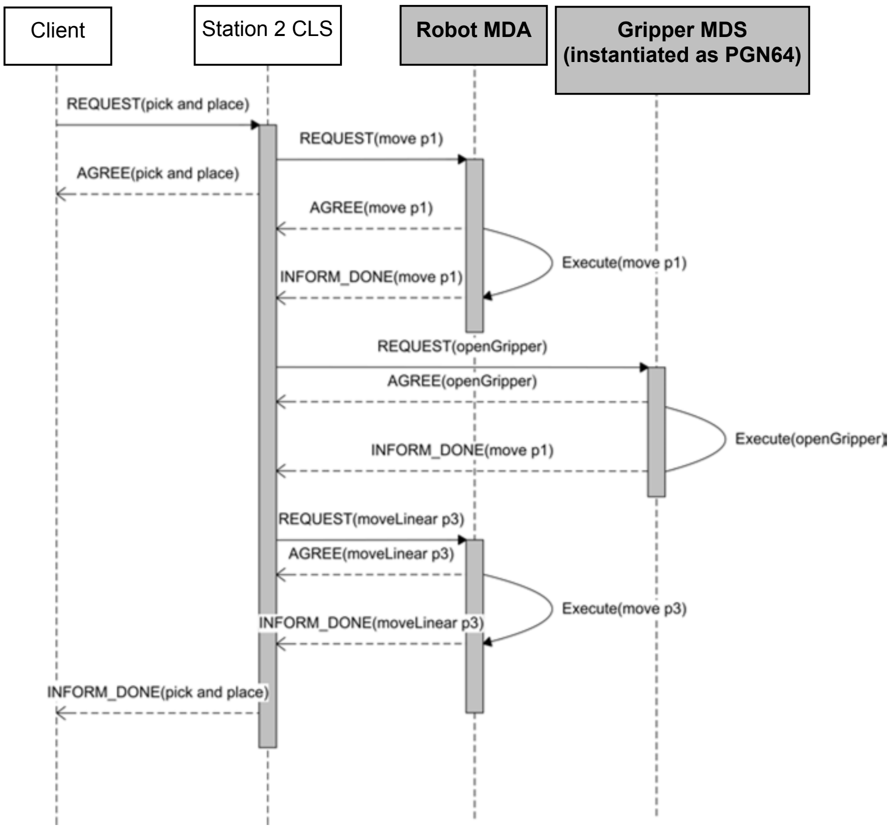


Fig. 16.7 The pick-and-place composed skill (only steps 1, 2 and 9 are represented)

16.7 Conclusions

It may seem odd to present a chapter with a strong focus on shop-floor agility in the context of dynamic supply chains. However, in order for a company to successfully participate in such an endeavour overall agility is required. Examining the system as a whole (dynamic supply chains and participating enterprises) one can verify that the main challenges and requirements propagate from the highest to the lowest (device) levels. In other words, structurally the whole system is self-similar at distinct abstraction levels. Agility at shop-floor level is, in this context, an elementary property. The requirements are tougher in the case of even more dynamic and complex organizations, such as the ones devised under the umbrella of collaborative networks.

The last sections of this chapter clearly show that, through the proof of concept demonstrator, shop-floor agility can be achieved and supported by existing and evolving standards. The experimental usage of a DPWS stack has provided adequate support and performance. Given its reduced footprint its applicability domain extends to embedded tiny devices supporting the idea of a truly ubiquitous and intelligent ecosystem.

It is worth recalling that technology alone is only a facilitator. Successful implementation requires systematic design, assessment and application of abstractions/models and architectures such as the one described.

References

Andrews T, Curbera F et al (2003) Business process execution language for web services, Version 11

Arkin, A, Askary S et al (2002) Web service choreography interface (WSCI) 10 W3C Note

Barata J, Camarinha-Matos LM (2000) Shop-floor reengineering to support agility in virtual enterprise evironments. In: L M Camarinha-Matos, H Afsarmanesh and R Rabelo (eds) E-business and virtual enterprises, Kluwer Academic Publishers, London, 1:287–291

Barata J, Camarinha-Matos LM et al (2001) Integrated and distributed manufacturing, a multi-agent perspective. In: Proceedings of 3rd Workshop on European Scientific and Industrial Collaboration, Enschede, The Netherlands

Barata J, Ribeiro L et al (2007a) Diagnosis using service oriented architectures (SOA). In: Proceedings of the International Conference on Industrial Informatics, Vienna

Barata J, Ribeiro L et al (2007b) Diagnosis on evolvable production systems. In: Proceedings of International Symposium on Industrial Electronics, Vigo

Booth D, Hass H et al (2004) Web services architecture. In W3C Working Group Note 11

Box D, Ehnebuske D et al (2000) Simple object access protocol (SOAP). In W3C Note 08

Camarinha-Matos LM (2002) Virtual organisations in manufacturing. In: Proceedings the 12th International Conference on Flexible Automation and Intelligent Manufacturing Munich, Oldenbourg: pp. 1036–1054

Camarinha-Matos LM, Barata J (2001) Contract-based approach for shop-floor re-engineering. In: R Bernhardt and HH Erbe (eds.) Cost oriented automation, Elsevier, Oxford, 1:141–148

Chan S, Conti D, Kaler C. et al. (2006) Devices profile for web services. http://schemas.xmlsoap.org/ws/2006/02/devprof/.

Davidow WH, Malone MS (1993) The virtual corporation: structuring and revitalizing the corporation for the 21st century, Harper Business, New York

FIPA (2002) FIPA Request Interaction Protocol Specification (SC00026H)

Goldman SL, Nagel RN et al (1995) Agile competitors and virtual organizations: strategies for enriching the customer, New York

Hormozi AM (2001) Agile manufacturing: the next logical step. Benchmarking Int. J. 8(2):132–143

Huhns MN, Singh MP (2005) Service-oriented computing: key concepts and principles. Internet Comput. 9(1):75-81

InLife (2006, 01/01/2007) Integrated ambient intelligence and knowledge based services for optimal life-cycle impact of complex manufacturing and assembly lines. http://wwwuninovapt/inlife/

Iung B, Levrat E et al (2007) E-maintenance: principles, review and conceptual framework. Cost effective automation in networked product development and manufacturing, Monterrey, NL México

Jammes F, Mensch A et al (2005) Service-oriented device communications using the device profile for web services. In: Proceedings of the 3rd international workshop on middleware for pervasive and ad-hoc computing
Jammes F, Mensch A et al (2007) Service-oriented device communications using the device profile for web services. In: Proceedings of 2007 Advanced Information Networking and Applications Workshop
Jammes F, Smit H (2005) Service-oriented architectures for devices – the SIRENA view. In: Proceedings of the International Conference on Industrial Informatics, Perth, Western Australia
Jardine AKS, Lin D et al (2006) A review on machinery diagnostics and prognostics implementing condition-based maintenance. Mech. Syst. Sign. Process., 20:1483–1510
Leitão P, Barata J et al (2001) Trends in agile and co-operative manufacturing cost oriented automation. In R Bernhardt and HH Erbe (eds), Elsevier, Oxford, 1:149–158
Lopes G, Pereira LM (2006) Prospective programming with ACORDA. Empirically successful computerized reasoning, Seattle, USA
Luck M, McBurney P et al (2005) AgentLinkIII – A road map for agent based computing (available in http://wwwagentlinkorg/)
Maskell, B (2001) The age of agile manufacturing. Supply Chain Manag.: Int. J. 6(1):5–11
Nagel R, Dove R (1992) 21st Century manufacturing enterprise strategy, Iacocca Institute, Lehigh University, USA
Onori, M (2002) Product design as an integral step in assembly system development. Ass. Autom. 22(3):203-205
Papazoglou M P, Traverso P et al (2005) Service-oriented computing: A research roadmap. Dagstuhl seminar proceedings 05462 (SOC)
Ribeiro L (2007) A aiagnostic infrastructure for manufacturing systems. Elect. Comput. Sci. Eng., New University of Lisbon MSC: 121
Ribeiro L, Barata J et al (2008a) MAS and SOA: complementary automation paradigms. In A Azevedo (ed) Innovation in Manufacturing Networks, Springer, Boston, pp. 259–268
Ribeiro L, Barata J et al (2008b) A generic communication interface for DPWS-based web services. In: Proceedings of the International Conference in Industrial Informatics INDIN, Daejeon, Korea
Ribeiro L, Barata J et al (2009a) Supporting agile supply chains using a service-oriented shop floor. Eng. Appl. Artif. Intell., 22(6):950–960
Ribeiro L, Barata J et al (2009b) Maintenance management and operational support as services in reconfigurable manufacturing systems. In IFAC Symposium on Information Control Problems in Manufacturing, Moscow
SIRENA (2006) Service infrastructure for real-time embedded network applications, from http://wwwsirena-iteaorg/Sirena/Homehtm
SOCRADES (2006) Service-Oriented Cross-layer infRAstructure for Distributed smart Embedded devices, from http://wwwsocradeseu/Home/defaulthtml
SODA (2006–2008) Service oriented device and delivery architecture. http://wwwsoda-iteaorg/Home/defaulthtml
Vernadat FB (1999) Research agenda for agile manufacturing. Int. J. Agile Mgmt. Syst. 1(1): 37–40

Index