BAB II
LANDASAN TEORI

2.1 Teori Dasar Sistem

2.1.1 Sistem

Menurut Nur Elfi Husda dan Yvonne Wangdra (2016, p.91), Sistem adalah suatu kumpulan dari komponen, unsur, variabel yang terorganisir, saling berinteraksi, saling tergantung satu sama lain, dan terpadu. Model umum sebuah sistem adalah *input*, proses, dan *output*. Hal ini adalah konsep sebuah sistem yang sangat sederhana, sebab sebuah sistem dapat memiliki beberapa masukan dan keluaran. Selain itu, sebuah sistem mempunyai karakteristik atau sifat-sifat tertentu yang mencirikan hal tersebut bisa dikatakan sebagai suatu sistem. Adapun karakteristik yang dimaksud adalah sebagai berikut:

1. Komponen Sistem *(Component)*

   Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, artinya saling bekerja sama membentuk satu kesatuan. Komponen-komponen sistem dapat berupa suatu subsistem. Setiap subsistem memiliki sifat dari sistem yang menjalankan suatu fungsi tertentu dan mempengaruhi proses sistem secara keseluruhan. Suatu sistem dapat memiliki sistem yang lebih besar atau sering disebut “supra sistem”.

2. Batasan Sistem *(Boundary)*

   Ruang lingkup sistem merupakan wilayah/ daerah yang membatasi antara sistem dengan sistem yang lain atau sistem dengan lingkung luarnya. Batasan sistem memungkinkan suatu sistem dipandang sebagai satu kesatuan yang tidak dapat terpisahkan.

3. Lingkungan Luar Sistem *(Environtment)*

   Bentuk apapun diluar ruang lingkup atau batasan sistem yang mempengaruhi operasi sistem disebut lingkungan luar sistem. Lingkungan luar sistem bersifat menguntungkan dan juga bersifat merugikan sistem tersebut. Dengan demikian, lingkungan luar harus tetap dijaga dan

dipelihara. Lingkungan luar yang merugikan harus dikendalikan, kalau tidak akan mengganggu kelangsungan hidup sistem tersebut.

4. Penghubung Sistem *(Interface)*

   Media yang menghubungkan sistem dengan subsistem lain disebut penghubung sistem atau *interface*. Penghubung memungkinkan sumber daya mengalir dari satu subsistem ke subsistem lain. Bentuk keluaran dari subsistem akan menjadi masukan untuk subsistem lain melalui penghubung tersebut. Dengan demikian, dapat terjadi suatu integrasi sistem yang membentuk satu kesatuan.

5. Masukan Sistem *(Input)*

   Energi yang dimasukkan kedalam sistem adalah masukkan sistem, yang dapat berupa pemeliharaan *(maintenance input)* dan sinyal *(signal input)*. Contoh di dalam suatu unit sistem komputer, "Program" adalah *maintenance* input yang digunakan untuk mengoperasikan komputernya dan "Data" adalah *signal input* untuk diolah menjadi informasi.

6. Keluaran Sistem *(Output)*

   Hasil energi yang diolah menjadi keluaran yang berguna. Keluaran ini merupakan masukan untuk subsistem yang lain seperti sistem informasi, Keluaran yang dihasilkan adalah informasi. Informasi dapat digunakan sebagai masukan dalam pengambilan keputusan atau hal-hal lain yang menjadi *input* bagi subsistem lain.

7. Pengolahan Sistem *(Process)*

   Suatu sistem memiliki proses yang akan mengubah masukan menjadi keluaran, contohnya adalah sistem akuntansi, dimana sistem akan mengolah data transaksi menjadi laporan-laporan yang dibutuhkan oleh pihak manajemen.

8. Sasaran Sistem *(Objective)*

   Suatu sistem memiliki tujuan, sasaran yang pasti, dan bersifat *deterministic*. Kalau sistem tidak memiliki sasaran maka operasi sistem tidak ada gunanya, suatu sistem dikatakan berhasil dapat mengenai sasaran atau tujuan yang telah direncanakan.

2.1.2 Informasi

Menurut Nur Elfi Husda dan Yvonne Wangdra (2016, p.94), Informasi adalah data yang telah diklasifikasi untuk digunakan dalam proses pengambilan keputusan. Sistem pengolahan informasi yaitu mengolah data dari tidak berguna menjadi berguna bagi penerimanya. Nilai informasi berhubungan dengan keputusan, maka informasi tidak diperlukan seperti keputusan berulang sederhana sampai keputusan strategis jangka panjang, nilai informasi paling berarti dalam konteks sebuah keputusan.

2.1.3 Sistem Informasi

Menurut Nur Elfi Husda dan Yvonne Wangdra (2016, p.96), Sistem informasi adalah sistem didalam suatu organisasi yang dimana mempertemukan kebutuhan pengolahan transaksi harian dalam mendukung fungsi operasi organisasi yang bersifat manajerial dengan kegiatan strategi suatu organisasi untuk menyediakan kepada pihak luar tertentu dengan laporan-laporan yang diperlukan.

2.1.4 Perancangan

Menurut Rosa A.S. dan M. Shalahuddin (2016, p.23), Perancangan adalah proses mendefinisikan suatu model atau rancangan perangkat lunak (*software*) dengan menggunakan teknik dan prinsip tertentu hingga model atau rancangan dapat diwujudkan menjadi perangkat lunak.

2.2 *Knowledge Management*

2.1.1 Definisi *Knowledge*

Menurut Sapri Wiguna Putra (2016) *knowledge* atau pengetahuan merupakan bagian dalam kehidupan sosial manusia modern saat ini, yang dimana pada era globalisasi ini diwarnai dengan maraknya inovasi yang ditandai dengan pesatnya perkembangan ilmu pengetahuan.

2.1.1.1 Tipe *Knowledge*

Menurut Suparto Darudiato dan Kevin Setiawan (2013) pengetahuan dapat dibagi dalam tiga jenis, yaitu:

a. *Tacit Knowledge*

   Tacit Knowledge merupakan pengetahuan yang dimiliki oleh seseorang dan sangat sulit untuk diformalisasikan, sulit dikomunikasikan atau dibagi dengan orang lain. Pemahaman yang melekat di dalam pengetahuan individu tersebut masih bersifat subjektif. Pengetahuan yang dimiliki oleh individu tersebut masih dapat dikategorikan sebagai intuisi dan dugaan. *Tacit knowledge* ini berada dan berakar di dalam tindakan maupun pengalaman seseorang.

b. *Explicit Knowledge*

   Explicit knowledge merupakan suatu pengetahuan yang telah berhasil terdokumentasikan, yang memiliki suatu sifat struktural, sistematis dan mudah untuk dikomunikasikan dan dibagikan kepada orang lain. Pengetahuan ini dapat berupa buku, jurnal, karya ilmiah, referensi atau lainnya.

c. *Potential Knowledge*

   Merupakan suatu pengetahuan yang digunakan untuk melakukan suatu analisis data dan mengubah data menjadi sebuah pengetahuan. Pengetahuan ini didapatkan dan berkembang dari hasil analisis terhadap data yang ada.

2.1.1.2 Siklus *Knowledge*

Menurut Sapri Wiguna Putra (2016) siklus *knowledge* terdiri dari:

a. Sosialisasi

   Merupakan proses *sharing* dan penciptaan *tacit knowledge* melalui interaksi dan pengalaman langsung.

b. Externalisasi

Merupakan pengartikulasian *tacit knowledge* menjadi *explicit knowledge* melalui proses dialog dan refleksi.

c. Kombinasi

Merupakan proses konversi *explicit knowledge* menjadi *explicit knowledge* yang baru melalui sistemisasi dan pengaplikasian *explicit knowledge* dan informasi.

d. Internalisasi

Merupakan proses pembelajaran dan akusisi *knowledge* yang dilakukan oleh anggota organisasi terhadap *explicit knowledge* yang disebarkan ke seluruh organisasi melalui pengalaman sendiri sehingga menjadi *tacit knowledge* anggota organisasi.

2.1.2 Definisi *Knowledge Management*

Menurut Satrio Dirgantoro (2014) *knowledge management* merupakan sistem yang dibuat untuk menciptakan, mendokumentasikan, menggolongkan, dan menyebarkan pengetahuan dalam organisasi sehingga pengetahuan mudah digunakan kapanpun dan oleh siapa siapapun.

2.1.3 Proses Inti *Knowledge Management*

Menurut Sapri Wiguna Putra (2016) proses inti *knowledge management* terdiri dari:

a. *Knowledge Goal*

Merupakan tujuan akhir dari *knowledge*. Dengan adanya *knowledge goal* ini maka komunitas akan lebih mudah untuk menentukan arah dan strategi guna mencapai tujuan.

b. *Knowledge Assessment*

Merupakan penilaian dari proses inti *knowledge management* di komunitas. Hal ini dilakukan untuk mengetahui sejauh mana hasil/ pengaruh yang telah dihasilkan *knowledge* di komunitas.

c. *Knowledge Identification*

Merupakan suatu proses pengidentifikasian *knowledge* baik dalam bentuk *tacit* maupun *explicit*. Proses ini diharapkan agar sekolah mampu mengetahui sejauh mana *knowledge* yang telah dimiliki. Dengan adanya identifikasi yang jelas maka diharapkan masing-masing individu dapat melakukan kegiatan yang optimal.

d. *Knowledge Acquisition*

Dengan adanya *knowledge acquisition*, diharapkan komunitas dapat menambah *knowledge* yang tadinya tidak dimiliki oleh sekolah, serta melengkapi *knowledge* awal yang dimiliki komunitas. Adapun *knowledge acquisition* biasanya bersumber dari luar komunitas. Misalnya, bekerjasama dengan organisasi, komunitas lain, dan lain-lain.

e. *Knowledge Development*

Inti dari *knowledge development* adalah menghasilkan *knowledge* baru dari *knowledge* yang ada sebelumnya sehingga akan berguna bagi sekolah dalam meningkatkan kualitas karyawan, guru ataupun kinerjanya.

f. *Knowledge Sharing* dan *Distribution*

Mengubah knowledge yang bersifat individu menjadi lebih kolektif. Dengan adanya *Knowledge Sharing* dan *Distribution* ini kualitas individu akan meningkat. Dan kegiatan ini tentunya membutuhkan fasilitas dan waktu yang cukup bagi individu yang terlibat.

g. *Knowledge Utilization*

Inti dari *knowledge management* adalah memastikan bahwa *knowledge* yang sudah ada di komunitas dipakai secara produktif untuk pengembangan dari sekolah tersebut.

h. *Knowledge Retention*

Knowledge yang ada di dalam individu maupun komunitas tidaklah didapatkan secara otomatis. Oleh karena itu, penyimpanan *knowledge* sangatlah penting agar *knowledge* dapat tersimpan dengan baik. Untuk itu, hal yang perlu diperhatikan dalam penyimpanan *knowledge* adalah media tempat penyimpanan *knowledge* tersebut.

2.3 *World Wide Web* (WWW)

Menurut Nur Elfi Husda dan Yvonne Wangdra (2016, p.128), *World Wide Web* (WWW) lebih dikenal dengan web merupakan sistem dimana informasi dalam bentuk teks, suara, gambar, dan lain-lain yang disimpan di *server-server* yang ada di seluruh dunia. Dokumen web dibuat dengan menggunakan format HTML.

2.4 HTTP

Menurut Priyanto Hidayatullah dan Jauhari Khairul Kawistara (2015, p.5), HTTP (*Hypertext Transfer Protocol)* adalah suatu protokol agar *client* dan *server* bisa berkomunikasi dengan gaya *request-response*. HTTP menentukan bagaimana format pesan dan bagaimana cara pengirimannya, serta bagaimana *web server* dan *browser* beraksi dan bereaksi terhadap berbagai perintah.

2.5 *Web Browser*

Menurut Nur Elfi Husda dan Yvonne Wangdra (2016, p.129), *web browser* adalah software yang digunakan untuk mencari dan menampilkan dokumen *web* dalam format HTML. Dengan browser, para pengguna komputer dapat mencari dan menelusuri serta melihat isi dari dokumen *web* dan berpindah dari sebuah tempat ke tempat lain di *web*. Contoh program browser yang populer misalnya internet explorer, netscape, opera, Mozilla, dan lain-lainnya.

2.6 PHP

Menurut Priyanto Hidayatullah dan Jauhari Khairul Kawistara (2015, p.231), PHP merupakan suatu bahasa *scripting* khususnya digunakan untuk *web development.* Karena sifatnya yang *server side scripting*, maka untuk menjalankan PHP harus menggunakan web server. PHP dapat diintegrasikan dengan HTML, JavaScript, JQuery, Ajax, biasanya PHP lebih banyak digunakan bersamaan dengan *file* bertipe HTML.

2.7 URL (*Universal Resource Locator*)

Menurut Nur Elfi Husda dan Yvonne Wangdra (2016, p.129), URL (*Universal Resource Locator*) merupakan konsep nama *file* standar yang diperluas dengan jaringan untuk menentukan lokasi informasi pada *web browser*. Setiap protokol bahasa (HTTP, Telnet, FTP, dll) mempunyai sistem penulisan alamat yang berbeda-beda, contoh URL: (http://www.facebook.com).

2.8 Teori Pemgembangan Sistem

2.8.1 *Unified Modeling Language* (UML)

Menurut Rosa A.S. dan M. Shalahuddin (2016, p.133), *Unified Modelling Language* (UML) adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak, UML menawarkan sebuah standar untuk merancang model sebuah sistem. Tujuan Penggunaan UML yaitu untuk memodelkan suatu sistem yang menggunakan konsep berorientasi objek dan menciptakan bahasa pemodelan yang dapat digunakan baik oleh manusia maupun mesin. Tipe-tipe Diagram UML adalah sebagai berikut:

a. *Use Case Diagram*

   Use Case Diagram adalah gambar dari beberapa atau seluruh aktor dan *use case* dengan tujuan untuk mengenali interaksi mereka dalam sistem. *Use case diagram* menggambarkan fungsionalitas dari sebuah sistem, yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case* mempresentasikan sebuah interaksi antara *actor* dan sistem. Komponen-komponen relasi dalam *use case* antara lain:

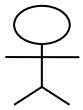
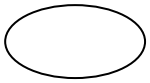
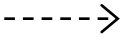
   1. *Assocciation,* Menghubungkan *link* antar element.
   2. *Generaliziont,* Disebut juga *interitance* (pewaris), sebuah elemen dapat spesialis dari elemen lainnya.
   3. *Depedency,* Sebuah element bergantung dalam beberapa cara ke elemen lainnya.
   4. *Aggregation,* Bentuk *association* dimana sebuah elemen berisi elemen lainnya.

Tipe relasi/ *stereotype* yang mungkin terjadi pada *use case* diagram:

1. <<*include*>>, Yaitu kelakuan yang harus terpenuhi agar sebuah *event* dapat terjadi, dimana pada kondisi ini sebuah *use case* adalah bagian dari *use case* lainnya.
2. <<*extends*>>, kelakuan yang hanya berjalan di bawah kondisi tertentu seperti menggerakan *alarm*.
3. <<*communicates*>>, mungkin ditambahkan untuk asosiasi yang menunjukkan asosiasinya adalah *communicates association*. Ini merupakan pilihan selama asosias tipe *relationship* yang dibolehkan antara *actor* dan *use case.*

Dalam *use case* diagram terdapat istilah seperti aktor, *use case* dan *case relationship.* Penjelasan simbol pada tabel 2.1

Tabel 2.1 Simbol *Use Case*

Simbol	**Keterangan**
	**Aktor** : Seseorang atau sesuatu yang berinteraksi dengan sistem yang sedang dikembangkan.
	***Use case*** : perangkat tertinggi dari fungsionalitas yang dimiliki sistem.
	Association *:* relasi antara actor dan *use case*
	Generalisasi*:* untuk memperlihatkan struktur pewaris yang terjadi.
	***Include*** **:** menspesifikasikan bahwa *use case* sumber secara *eksplisit.*
	***Collaboration :*** interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).

Tabel 2.1 (Lanjutan)

<----------	***Extend :*** menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.

b. *Activity Diagram*

Activity Diagram menggambarkan rangkaian aliran dari aktifitas yang mana digunakan untuk mendeskripsikan aktivitas yang dibentuk dalam suatu operasi sehingga dapat digunakan untuk aktivitas lainnya, seperti *use case* atau interaksi. *Activity Diagram* berupa *flowchart* yang digunakan untuk memperlihatkan aliran kerja dari sistem. Notasi yang digunakan dalam *Activity Diagram* adalah sebagai berikut:

a. *Activity*

Notasi yang menggambarkan pelaksanaan dari beberapa proses dalam aliran pekerjaan.

b. *Transition*

Notasi yang digunakan untuk memperlihatkan aliran kontrol dari *activity* ke *activity*.

c. *Decision*

Notasi yang menandakan kontrol cabang aliran berdasarkan *decision point.*

d. *Synchonization bars*

Aliran kerja notasi ini menandakan bahwa beberapa aktivitas dapat diselesaikan secara bersamaan.

Komponen-komponen dari *Activity Diagram* dapat dilihat pada tabel 2.2 berikut:

Tabel 2.2 Simbol *Activity Diagram*

No	Gambar	Nama	Keterangan
1		*Actifity*	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain.
2		*Action*	State dari sistem yang mencerminkan eksekusi dari suatu aksi.
3		*Initial Node*	Bagaimana objek dibentuk atau diawali.
4		*Actifity Final Node*	Bagaimana objek dibentuk dan dihancurkan.
5		*Fork Node*	Satu aliran pada tahap tertentu berubah menjadi beberapa aliran.

c. *Sequence Diagram*

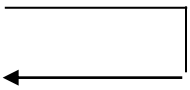
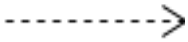
Sequence Diagram menggambarkan kolaborasi dinamis untuk menunjukkan rangkaian pesan yang dikirim antar objek, interaksi antar objek, dan sesuatu yang terjadi pada titik tertentu dalam eksekusi sistem. *Sequence Diagram* menjelaskan interaksi objek yang disusun berdasarkan urutan waktu. Dalam *Sequence Diagram* terdapat 2 simbol yaitu:

a. *Actor*, untuk menggambarkan pengguna sistem.

b. Lifeline, untuk menggambarkan kelas dan objek.

Komponen-komponen dari S*equence Diagram* dapat dilihat pada tabel 2.3 berikut:

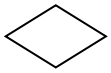
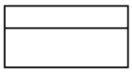
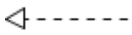
Tabel 2.3 Simbol *Sequence Diagram*

NO	GAMBAR	NAMA	KETERANGAN
1		*LifeLine*	Objek *entity*, antarmuka yang saling berinteraksi.
2		*Message*	Spesifikasi komunikasi antar objek yang memuat informasi tentang aktivitas yang terjadi.
3		*Messege to self*	Menggambarkan pesan yang menuju dirinya sendiri.
4		*Return Message*	Menggambarkan pengembalian dari pemanggilan prosedur.

d. *Class Diagram*

Class Diagram menggambarkan struktur data dan deskripsi *class, package,* dan objek beserta hubungan satu sama lain. *Class Diagram* berfungsi untuk menjelaskan tipe dari objek sistem dan hubungannya dengan objek yang lain. *Class* memiliki 3 area pokok yaitu nama, atribut, dan metode. *Class* menggambarkan keadaan (*attribute/ property*) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut. Komponen-komponen dari *Class Diagram* dapat dilihat pada tabel 2.4 berikut:

Tabel 2.4 Simbol *Class Diagram*

No	Gambar	Nama	Keterangan
1		*Generalization*	Hubungan objek anak *(descendent)* berbagi perilaku dari objek yang ada di atasnya objek induk (*ancestor*).
2		*Nary Association*	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
3		*Class*	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
4		*Collaboration*	Deskripsi dari urutan aksi-aksi ditampilkan sistem yang menghasilkan suatu hasil terukur bagi suatu *actor*.
5		*Realization*	Operasi yang dilakukan oleh suatu objek.
6		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempegaruhi elemen yang bergantung pada elemen yang tidak mandiri.

Tabel 2.4 (Lanjutan)

7		*Association*	yang menghubungkan antara objek satu dengan objek lainnya.

2.8.2 Metode *Rational Unified Process* (RUP)

Menurut Rosa A.S. dan M. Shalahuddin (2016, p.124), *Unified Process* atau dikenal proses iteratif dan *incremental* merupakan sebuah proses pengembangan perangkat lunak yang dilakukan secara iteratif (berulang) dan *incremental* (bertahap dengan proses menaik). Iteratif bisa dilakukan di dalam setiap tahap atau iteratif tahap proses pengembangan perangkat lunak untuk menghasilkan perbaikan fungsi yang inkremental (bertambah menaik) dimana setiap interasi akan memperbaiki interasi berikutnya, salah satu *Unified Process* yang terkenal adalah RUP (*Rational Unified Process*). RUP (*Rational Unified Process*) adalah pendekatan pengembangan perangkat lunak yang dilakukan berulang-ulang (*iterative*), fokus pada arsitektur (*architecture-centric*), lebih diarahkan berdasarkan pengguna kasus (*use case driven*). RUP merupakan proses perangkat lunak dengan pendefinisian yang baik (*well defined*) dan perstrukturan yang baik (*well struktured*).

a. Kelebihan RUP

Pendekatan iteratif/ pengulangan dari RUP dapat mengokomodir beberapa kelemahan pengembangan perangkat lunak tanpa menggunakan konsep pengulangan.

1. RUP mengakomodasikan perubahan kebutuhan perangkat lunak, Kebutuhan untuk mengubah dan menambah fitur karena perubahan teknologi atau keinginan pelanggan (*customer*) merupakan salah satu kendala yang sering dialami pengembang perangkat lunak yang berimbas pada terlambatnya waktu penyelesaian perangkat lunak.

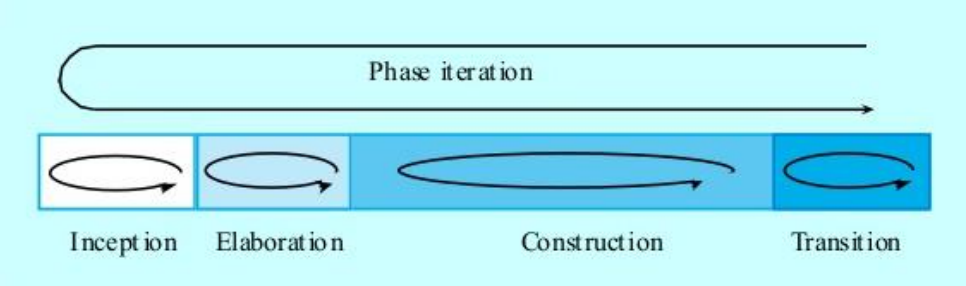
Keterlambatan ini dapat menyebabkan ketidakpuasan pelanggan (*customer*) dan pengembang menjadi frustasi.

2. Integrasi bukanlah sebuah proses besar dan cepat (*big bag*) diakhir proyek. Menempatkan integrasi dibagian akhir proyek biasanya memakan waktu yang cukup banyak, bisa mencapai 40 persen dari waktu proyek. Untuk mencegah hal ini maka pendekatan secara iteratif (pengulangan) dapat memecah proyek menjadi bagian iterasi kecil yang diakhiri dengan integrasi kecil, yang dimana nantinya digabungkan menjadi integrasi besar.
3. Risiko biasanya ditemukan atau dialamatkan selama pada proses intergrasi awal. Pendekatan intergrasi pada RUP mengurangi risiko pada iterasi awal dimana saat semua komponen diuji.
4. Manajemen berarti membuat perubahan taktik pada produk. Taktik produk misalnya pengembangan dengan waktu singkat akan menghasilkan produk dengan fungsi yang terbatas dan dapat cepat digunakan oleh *user* sehingga memperkenalkan produk lebih cepat kemasyarakat dibandingkan produk kompetitor lain yang masih sedang dikembangkan.
5. Mendukung fasilitas penggunaan kembali lebih mudah untuk mengidentifikasi bagian umum yang sering digunakan dalam aplikasi, dimana jika diimplementasikan secara iterasi dari pada mengidentifikasi pada saat perencanaan saja.
6. Kecacatan dapat ditemukan dan dapat diperbaiki pada beberapa iterasi dan menghasilkan arsitektur yang baik dan aplikasi berkualitas tinggi, kecacatan dideteksi lebih baik mulai pada iterasi awal daripada pada tahap pengujian.
7. Lebih baik menggunakan “anggota proyek” dibandingkan susunan secara seri pada tim proyek.
8. Anggota tim belajar selama proyek berjalan, dimana anggota proyek memiliki peluang belajar dari kesalahan selama siklus pengembangan perangkat lunak dan memperbaiki kesalahan pada interasi berikutnya.

9. Pengembangan perangkat lunak dapat diperbaiki seiring proses pengembangan perangkat lunak setiap akhir proses iterasi, tidak hanya ada peninjauan mengenai target produk tetapi juga peninjauan pada kesalahan yang terjadi pada iterasi sebelumnya agar dapat diperbaiki pada iterasi berikutnya.

b. Fase RUP

RUP memiliki empat buah fase yang dapat dilakukan secara iteratif. Berikut ini adalah alur hidup RUP yang digambarkan pada gambar 2.2



Gambar 2.2 Alur Hidup RUP

Berikut ini penjelasan untuk setiap fase pada RUP.

1. *Inception* (Permulaan)

   Tahap ini lebih pada pemodelan proses bisnis yang dibutuhkan (*business modeling*) dan mendefinisikan kebutuhan akan sistem yang akan dibuat (*requiretments*). Berikut adalah tahap yang dibutuhkan pada tahap ini:

   a. Memahami ruang lingkup dari proyek (termasuk pada biaya, waktu, kebutuhannya, risiko dan lain sebagainya).
   b. Membangun kasus bisnis yang dibutuhkan.

   Hasil yang diharapkan dari tahap ini adalah memenuhi *lifecycle objective milestone* (batasan/ tonggak objektif dari siklus) dengan kriteria berikut:

   a. Umpan balik dari pendefinisian ruang lingkup, perkiraan biaya, dan perkiraan jadwal.

b. Kebutuhan dimengerti dengan pasti (dapat dibuktikan) dan sejalan dengan kasus primer yang dibutuhkan.
c. Kredibilitas dari perkiraan biaya, perkiraan jadwal, penentuan skala prioritas, risiko, dan proses pengembangan.
d. Ruang lingkup purwarupa (*prototype*) yang akan dikembangkan.
e. Membangun garis dasar dengan membandingkan perencanaan aktual dengan perencanaan yang direncanakan.

Jika akhirnya tahap ini yang diinginkan tidak dicapai maka dapat dibatalkan atau diulang kembali setelah dirancang ulang agar kriteria yang diinginkan dapat dicapai. Batas/ tonggak objektif digunakan untuk mendeteksi apakah sebuah kebutuhan akan sistem dapat diimplementasi atau tidak.

2. *Elaboration* (Perluasan/ Perencanaan)

Tahap ini lebih difokuskan pada perencanaan arsitektur sistem. Tahap ini juga dapat mendeteksi apakah arsitektur sistem yang diinginkan dapat dibuat atau tidak. Mendeteksi risiko yang mungkin terjadi dari arsitektur yang dibuat. Tahap ini lebih pada analisis dan desain sistem serta implementasi sistem yang fokus pada purwarupa sistem *(prototype)*. Hasil yang diharapkan dari tahap ini adalah memenuhi *lifecycle architecture milestone* (batas/ tonggak arsitektur dari siklus) dengan kriteria berikut:

a. Model kasus yang digunakan *(use case)* dimana kasus dan aktor yang terlihat telah didefinisikan dari sebagian besar kasus harus dikembangkan. Model *use case* harus 80 persen lengkap dibuat.
b. Deskripsi dari arsitektur perangkat lunak dari proses pengembangan sistem perangkat lunak telah dibuat.
c. Rancangan arsitektur yang dapat diimplementasikan dan mengimplementasikan *use case.*
d. Kasus bisnis dan proses bisnis dan daftar risiko yang sudah mengalami perbaikan (revisi) telah dibuat.
e. Rencana pengembangan untuk seluruh proyek telah dibuat.

f. Purwarupa *(prototype)* yang dapat didemonstrasikan untuk mengurangi setiap risiko teknis yang didefinisikan.

Jika pada tahap ini tidak dicapai maka dapat dibatalkan atau diulang kembali. Batas/ tonggak arsitektur digunakan untuk mendeteksi apakah sebuah kebutuhan akan sistem dapat diimplementasikan atau tidak melalui pembuatan arsitektur.

3. *Construction* (Konstruksi)

   Tahap ini fokus pada pengembangan komponen dan fitur-fitur sistem. Tahap ini lebih pada implementasi dan pengujian sistem yang fokus pada implementasi perangkat lunak pada kode program. Tahap ini menghasilkan produk perangkat lunak dimana menjadi syarat dari *Intial Operational Capability Milestone* atau batas/ tonggak kemampuan operasional awal.

4. *Transition* (Transisi)

   Tahap ini lebih pada *deployment* atau instalasi sistem agar dapat dimengerti oleh *user*. Tahap ini menghasilkan produk perangkat lunak dimana menjadi syarat dari *Initial Operational Capability Milestone* atau batas/ tonggak kemampuan operasional awal. Aktivitas pada tahap ini termasuk pada pelatihan *user*, pemeliharaan dan pengujian sistem apakah sudah memenuhi harapan u*ser*. Produk perangkat lunak juga disesuaikan dengan kebutuhan yang didefinisikan pada tahap *inception.* Jika semua kriteria objektif terpenuhi maka dianggap sudah memenuhi *Products Release Milestone* (batas/ tonggak peluncuran produk) dan pengembangan perangkat lunak selesai dilakukan. Akhir dari keempat fase ini adalah produk perangkat lunak yang sudah lengkap. Keempat fase pada RUP dijalankan secara berurutan dan iteratif dimana setiap iterasi dapat digunakan untuk memperbaiki iterasi berikutnya.

2.9 Database

Menurut Nur Elfi Husda dan Yvonne Wangdra (2016, p.119), Basis data adalah sekumpulan data yang terhubung satu sama lain secara logika dan suatu deskripsi data yang dirancang untuk memenuhi kebutuhan informasi suatu organisasi.

Pendefinisian basis data meliputi spesifikasi dari tipe data, struktur dan batasan dari data atau informasi yang akan disimpan. *Database* merupakan salah satu komponen yang penting dalam sistem informasi, karena merupakan basis dalam menyediakan informasi para pengguna atau *user*.

2.10 Perangkat Lunak Pendukung

2.10.1 *Adobe dreamweaver cs 4*

Menurut Wahana Komputer (2010, p.2), *Adobe Dreamweaver cs 4* adalah sebuah HTML editor profesional untuk mendesain *web* secara visual dan mengelola situs atau halaman *web*. *Adobe Dreamwever cs 4* memiliki beberapa kemampuan yang bukan hanya sebagai *sofware* untuk desain *web* saja, tetapi juga menyunting kode serta pembuatan aplikasi *web*, antara lain: JSP, PHP, ASP, XML, dan *ColdFusion*. *Adobe Dreamweaver cs 4* merupakan software utama yang digunakan oleh *web designer* dan *web programmer* dalam mengembangkan suatu situs *web*.

2.10.2 XAMPP

Menurut Muhammad Imansyah (2010, p.4), *XAMPP* merupakan aplikasi server yang menggabungkan beberapa aplikasi server yang biasa digunakan di *web server*. Berikut beberapa komponen-komponen yang terdapat pada XAMPP, yaitu *Apache* (*web server*), *MySQL* (*database server*), *Filezila* FTP *server, Mercury* Mail (mail *server*), *phpMyAdmin* (*web-based interface* MySQL).

2.10.3 StarUML

Menurut Evi Triandini dan Gede Suardika (2012, p.1), *StarUML* adalah sebuah proyek *open source* untuk pengembangan secara cepat, fleksibel, *extensible, featureful*, dan bebas-tersedia. Tujuan dari proyek *StarUML* adalah untuk membangun sebuah alat pemodelan perangkat lunak dan juga *platform* yang menarik dalam mendukung UML (*Unified Modeling Language*). Berdasarkan pada UML version 1.4 dan dilengkapi 11 macam diagram yang berbeda, selanjutnya mendukung notasi UML 2.0 dan juga

mendukung pendekatan MDA (*Model Driven Architecture*) dengan dukungan konsep UML. *StarUML* dapat memaksimalkan pruduktivitas dan kualitas dari suatu *software project*.

2.11 *Black Box Testing*

Menurut Rosa A.S. dan M. Shalahuddin (2016, p.275), *Black box testing* adalah pengujian perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan. Pengujian *black box* dilakukan dengan membuat kasus uji yang bersifat mencoba semua fungsi dengan memakai perangkat lunak, kasus uji yang dibuat untuk melakukan pengujian *black box* dibuat dengan kasus benar dan kasus salah, misalkan untuk kasus proses login maka kasus uji yang dibuat adalah:

a. Jika *user* memasukkan nama pemakai (*username*) dan kata sandi (*password*) benar.
b. Jika *user* memasukkan nama pemakai (*username*) dan kata sandi (*password*) yang salah, misalnya nama pemakai benar tetapi kata sandi salah, atau sebaliknya, atau keduanya salah.