BAB II
LANDASAN TEORI

2.1 Sistem

Sistem adalah kumpulan dari elemen-elemen yang berinteraksi untuk mencapai suatu tujuan tertentu dari, definisi diatas dapat disimpulkan bahwa sistem merupakan suatu alat untuk menghubungkan suatu jaringan kerja dari sekumpulan elmen-elmen yang saling berinteraksi antara satu dengan yang lainnya, sehingga mencapai tujuan yang telah ditetapkan atau direncanakan. Sebagai salah satu contoh adalah sistem akuntansi dimana semua kegiatan yang dilakukan dari mulai pencatatan, pengelompokan, perangkuman dan pelaporan merupakan suatu elemen sistem yang saling berinteraksi adapun tujuan yang ditetapkan adalah laporan keuangan. *Output* dari suatu sistem adalah adanya informasi yang mendukung untuk mengambil keputusan, baik keputusan manajerial ataupun keputusan oprasional disetiap tingkatan menejemen menurut (Triyanto, 2018)

2.2 Penjadwalan

Penjadwalan adalah aktivitas perencanaan untuk menentukan kapan dan dimana setiap operasi sebagai bagian dari pekerjaan secara keseluruhan harus dilakukan pada sumber daya yang terbatas, serta pengalokasian sumber daya pada suatu waktu tertentu dengan memperhatikan kapasitas sumber daya yang ada. Penjadwalan dapat diartikan sebagai pengalokasian sejumlah sumber daya (resource) untuk melakukan sejumlah tugas atau operasi dalam jangka waktu tertentu dan merupakan proses pengambilan keputusan yang peranannya sangat penting dalam industri manufaktur dan jasa yaitu mengalokasikan sumber-sumber daya yang ada agar tujuan dan sasaran perusahaan lebih optimal. Menurut (Juniarahmatunisa, 2014) penjadwalan dapat didefinisikan sebagai proses pengalokasian sumber daya untuk mengerjakan sekumpulan tugas dalam jangka waktu tertentu dengan 2 arti penting sebagai berikut.

a. Penjadwalan merupakan suatu fungsi pengambilan keputusan untuk membuat atau menentukan jadwal.

b. Penjadwalan merupakan suatu teori yang berisi sekumpulan prinsip dasar, model, teknik, dan kesimpulan logis dalam proses pengambilan keputusan yang memberikan pengertian dalam fungsi penjadwalan. Penjadwalan dibutuhkan untuk mengurangi alokasi tenaga operator, mesin dan peralatan produksi, dan dari aspek lainnya untuk lebih efisien. Hal ini sangat penting dalam pengambilan keputusan.

2.3 Priority Scheduling

adalah suatu kasus khusus dari penjadwalan berprioritas. Tiap-tiap proses dilengkapi dengan nomor prioritas (integer). CPU dialokasikan untuk proses yang mempunyai prioritas paling tinggi (nilai integer terkecil biasanya merupakan prioritas terbesar). misalkan beberapa proses memiliki prioritas yang sama, maka akan digunakan algoritma FCFS.

Priority Scheduling ini juga memiliki dua skema :

1. non-preemptive
2. preemptive

   Jika ada proses P1 yang datang pada saat P0 sedang berjalan, maka yang akan dilihat adalah prioritas P1. Tapi seandainya prioritas P1 lebih besar dibanding dengan prioritas P0, maka :

1. Pada non-preemptive, algoritma tetap akan menyelesaikan P0 sampai habis CPU burst-nya, dan meletakkan P1 pada posisi head queue.
2. Sedangkan pada preemptive, P0 akan dihentikan dulu, dan CPU ganti dialokasikan untuk P1. Misalnya terdapat lima proses P1,P2,P3, P4 dan P5 yang datang secara berurutan dengan CPU burst dalam milidetik.

   Diantara satu proses dengan proses yang lain terdapat waktu tunggu, nah, untuk menghitung waktu tunggu tersebut mari kita perhatikan contoh berikut.

Contoh :

Tabel 2.1 Penjadwalan

z	Arrival Time (AT)	Burst Time (BT	Size (kb)
P1	0	10	100 kb
P2	2	8	150 kb
P3	3	12	175 kb
P4	5	5	100

2.4 *Sublime Text 3*

Sublime text adalah aplikasi editor untuk kode dan teks yang dapat berjalan diberbagai *platform operating system* dengan menggunakan teknologi phyton apk. terciptanya aplikasi ini terinspirasi dari aplikasi vim, aplikasi ini sangatlah fleksibel dan *powerfull sublime text* bukanlah aplikasi opensource dan juga aplikasi yang dapat digunakan dan didapatkan secara gratis, akan tetapi beberapa fitur pengembangan fungsionalitas (*packages*) dari aplikasi ini merupakan hasil dari temuan dan mendapat dukungan penuh dari komunitas serta memiliki linsensi aplikasi gratis, *Sublime text* mendukung berbagai bahasa pemrograman dan mampu menyajikan fitur *syntax highlight* hampir di semua bahasa pemrograman yang didukung ataupun dikembangkan oleh komunitas seperti; *c, c++, c#, css, d, dylan, erlang, html, groovy, haskell, java, javascript, latex, lisp, lua, markdown, matlab, ocaml, perl, php, python, r, ruby, sql, tcl, textile and xml*. Biasanya bagi bahasa pemrograman yang didukung ataupun belum terdukung secara default dapat lebih dimaksimalkan atau didukung dengan menggunakan *add-ons* yang bisa didownload sesuai kebutuhan user. (Faridl, 2015)

2.5 *Xampp*

Xampp Adalah perangkat lunak bebas, yang mendukung banyak system operasi, merupakan kompilasi dari beberapa program. Fungsinya adalahsebagai *server* yang berdiri sendiri (localhost), yang terdiri atas program *Apache HTTP Server*, *MySQL* database, dan penerjemah bahasa yang ditulis dengan bahasa pemrograman PHP dan Perl. Nama *XAMP* merupakan singkatan dari X (empat system operasi apapun), Apache, *MySQL, PHP* dan Perl. Program ini tersedia dalam GNU *General Public License* dan bebas, merupakan web *server* yang mudah digunakan yang dapat melayani tampilan halaman web yang dinamis di kutip dari (Priyanti, 2013)

2.6 Basis Data

Menurut (Rosa & Shalahudin, 2015) mengatakan basis data (*database)* adalah system terkomputerisasi yang tujuan utamanya adalah memelihara data yang sudah diolah atau informasi dan membuat informasi tersedia saat dibutuhkan.pada intinya basis data adalah media untuk menyimpan data agara dafapat diakses dengan mudah dan cepat.

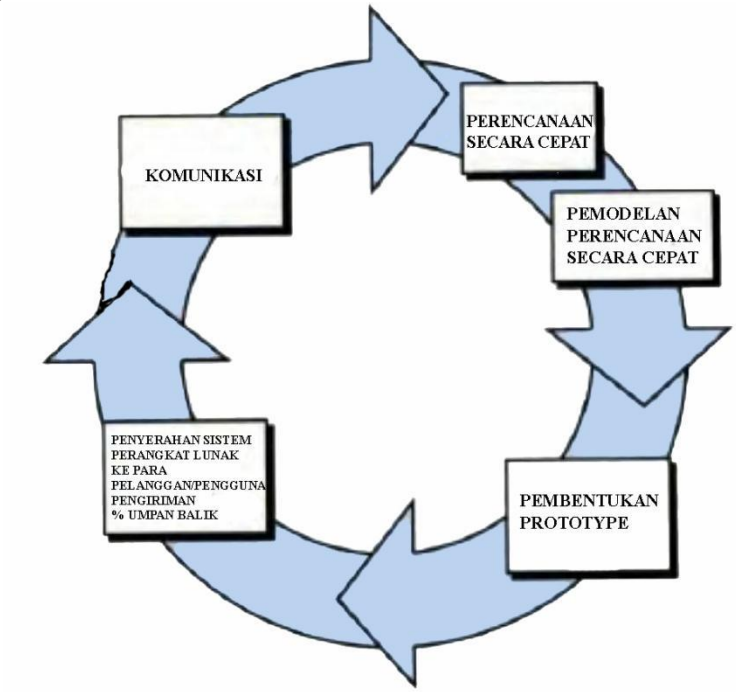
2.7 Metode Pengembangan Perangkat Lunak Menggunakan Metode *Prototype*

Metode Penelitian yang digunakan penulis didasarkan pada metode pengembangan perangkat lunak *prototype* yang memiliki 5 tahap.

Alasan penulis mengambil metode pengembangan sistem prototype karena pengguna dapat dengan mudah beradaptasi dengan sistem yang dikembangkan sesuai dengan kebutuhan pemakai. Selain itu pada saat tahap evaluasi sistem, jika Terdapat revisi pada sistem yang dirancang, programer tidak harus mengulang dari tahap awal

2.7.1 Metode *Prototype*

Pembuatan prototipe menurut (Pressman, 2012) dalam bukunya seringkali pelanggan mendefinisikan sejumlah sasaran perangkat lunak secara umum, tetapi tidak bisa mengidentifikasi spesifikasi kebutuhan yang rinci untuk fungsi-fungsi dan fitur-fitur yang nantinya yang akan dimiliki perangkat lunak yang akan dikembangkan.



Gambar 2.1 Metode *prototype*

Berikut penjelasan tahapan metode prototipe yang digunakan dalam penelitian ini.

1. Komunikasi

Pembuatan prototipe dimulai dengan dilakukannya komunikasi antara tim pengembang perangkat lunak dengan pada pelanggan.

2. Perencanaan Secara Cepat

Tim pengembang perangkat lunak akan melakukan pertemuan-pertemuan dengan para stakeholder untuk mendefinisikan sasaran keseluruhan untuk perangkat lunak yang akan dikembangkan, mengidentifikasi spesifikasi kebutuhan apapun yang saat ini diketahui, dan menggambarkan area-area dimana definisi jauh lebih pada iterasi selanjutnya merupakaan keharusan.

3. Pemodelan Perancangan Secara Cepat

Iterasi pembuatan prototipe direncanakan dengan cepat dan pemodelan (dalam bentuk"rancangan cepat") dilakukan, suatu rancangan cepat berfokus pada representasi semua aspek perangkat lunak yang akan terlihat oleh para pengguna akhir (misalnya rancangan antarmuka pengguna [*user interface*] atau format tampilan).

4. Pembentukan Prototipe

Rancangan cepat akan memulai kontruksi pembuatan prototipe, selanjutnya prototipe kemudian akan di serahkan kepada para *stakeholder* dan kemudian mereka akan melakukan evaluasi-evaluasi terhadap prototipe yang telah dibuat sebelumnya, kemudian akhirnya akan memberikan umpan-balik yang akan digunakan untuk memperhalus spesifikasi kebutuhan.

5. Penyerahan Sistem Perangkat Lunak Ke Para Pelanggan/Pengguna

Iterasi akan terjadi saat prototipe diperbaiki untuk memenuhi kebutuhan dari para *stakeholder*, sementara pada saat yang sama memungkinkan kita lebih memahami kebutuhan apa yang akan dikerjakan pada iterasi selanjutnya.

2.8 Unified Modeling Language (UML)

Menurut (Yasin, 2012)mendefinisikan *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 pengunaan uml yaitu untuk memodelkan suatu sistem yang menggunakan konsep berorientasi objek dan menciptakan bahasa pemodelan yang dapat digunakan baik oleh manusia maupun mesin.

Menurut verdi yasin (2012) tipe-tipe diagaram uml adalah sebagai berikut :

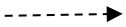
1) Use case diagram

Use case diagaram adalah gambar dari beberapa atau seluruh aktor dan *use case* dengan tujuan yang mengenali interaksi mereka dalam suatu sistem. *Use case diagram* menggambarkan fungsionalitas yang diharapkan dari sebuah sistem, yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case* mepresentasikan sebuah interaksi antara actor dan sistem.

Dalam *use case* diagram terdapat istilah seperti aktor, *use case* dan *case relationship.* Penjelasan simbol pada tabel 2.2

2.8.1 Simbol-simbol Use case

Tabel 2.2 Simbol Use Case

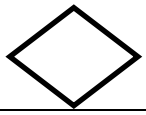
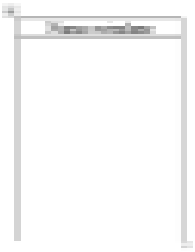
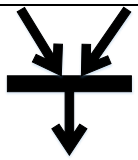
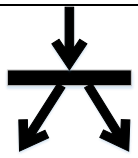
Simbol	Keterangan
	Aktor : seseorang atau sesuatu yang berinteraksi dengan sistem yang sedang dikembangkan.
	*Use case* : perungkat tertinggi dari fungsionalitas yang dimiliki sistem.
	*Association :*adalah relasi antara actor dan *use case*.
	*Generalisasi :* untuk memperlihatkan struktur pewaris yang terjadi.

2) Activity diagram

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

2.8.2 Simbol Activity Diagram

Tabel 2.3 Simbol *Activity Diagram*

Simbol	Keterangan
	*Activity* : memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain.
	*Initial node* : bagaimana objek dibentuk atau diawali
	*Actifity final node* : bagaimana objek dibentuk dan diakhiri.
	*Decision* : asosiasi percabangan dimana jika ada pilihan aktifitas lebih dari satu.
	*Swimlane* : memisalkan organisasi bisnis yang bertanggung jawab terhadap aktifitas yang terjadi.
	*Join* : digunakan untuk menunjukkan kegiatan yang digabungkan.
	*Fork* : digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel

3) *Sequence diagram*

Sequence diagram menggambarkan kolaborasi dinamis antara sejumalah dan untuk menunjukkan rangkaian pesan yang dikirim antar objek juga interaksi antar objek, 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

2.8.3 Simbol *Squense Diagram*

Tabel 2.4 Simbol *Sequense Diagram*

Simbol	**Deskripsi**
Kelas Nama_kelas +atribut +operasi( )	Kelas pada struktur system
Antarmuka/*interface*  **Nama_*interface***	Sama dengan konsep *interface* dalam pemrograman berorientasi objek.
Asosisasi/*association* ———	Relasi antarkelas dengan makna umum, asosiasi biasanya disertai dengan *multiplicity.*
Asosisasi berarah/*directed association* ——→	Relasi antarkelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*.

Generalisasi 	Relasi antarkelas dengan makna Generalisasi-spesialisasi (umum
Kebergantungan/*dependency* 	Relasi antarkelas dengan makna Kebergantungan antarkelas
Agregasi/*aggregation* 	Relasi antar kelas dengan makna semua-bagian

4) Class diagram

Class diagram menggambarkan dtruktur data dan desripsi *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.

2.9 Black Box Testing

Menurut (Rosa & Shalahudin, 2015) *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 apakah sesuai dengan spesifikasi yang dibutuhkan. Kasus uji yang dibuat untuk melakukan pengujian *black box* harus dibuat dengan kasus benar dan kasus salah, misalkan untuk kasus proses login maka kasus uji yang dibuat adalah :

1. Jika user memasukkan nama pemakai (*username*) dan kata sandi (*password*) benar.
2. 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.