BAB II
DASAR TEORI

2.1 Landasan Teori

2.1.1 Rekayasa Perangkat Lunak

Pengertian perangkat lunak (software) komputer adalah sekumpulan data elektronik yg disimpan dan diatur oleh komputer, data elektronik yg disimpan oleh komputer itu dapat berupa program atau instruksi yg akan menjalankan suatu perintah.[1] Dalam perangkat lunak tidak hanya berisi program, tetapi juga semua dokumentasi dan konfigurasi data yang saling terhubung sehingga program beroperasi dengan benar. Perangkat lunak cukup sering dibuat dan akhirnya tidak dipakai karena tidak memenuhi kebutuhan pelanggan atau bahkan karena masalah non-teknis seperti ketidakmauan pengguna untuk mengubah cara kerja dari manual ke otomatis, atau ketidakmampuan pengguna menggunakan komputer. Oleh karena itu, rekayasa perangkat lunak dibutuhkan agar perangkat lunak yang dibuat tidak hanya menjadi perangkat lunak yang tidak dipakai.

Rekayasa Perangkat Lunak (RPL) adalah disiplin ilmu yang membahas semua spek produksi perangkat lunak, mulai dari tahap awal spesifikasi sampai pemeliharaan sistem. Rekayasa perangkat lunak berhubungan dengan masalah praktis dalam memproduksi perangkat lunak. Rekayasa perangkat lunak lebih fokus pada praktik pengembangan perangkat lunak yang bermanfaat kepada pelanggan (*customer*).

Menurut Al Fatta, Pengujian sistem merupakan proses mengeksekusi sistem perangkat lunak untuk menentukan apakah sistem perangkat lunak tersebut cocok dengan spesifikasi sistem dan berjalan sesuai dengan lingkungan yang diinginkan [2]. Pengujian sistem sering diasosiasikan dengan pencarian *bug*, ketidaksempurnaan program, kesalahan pada baris program yang menyebabkan kegagalan pada eksekusi sistem perangkat lunak.

Menurut Pressman, dalam melakukan perancangan sistem yang akan dikembangkan dapat menggunakan *prototype* [3]. Metode ini cocok digunakan untuk mengembangkan sebuah perangkat yang akan dikembangkan kembali. Metode ini dimulai dengan pengumpulan kebutuhan pengguna, dalam hal ini pengguna dari perangkat yang

dikembangkan adalah pegawai. Kemudian membuat sebuah rancangan yang selanjutnya akan dievaluasi kembali sebelum diproduksi secara benar.

Prototype bukanlah sesuatu yang lengkap, tetapi sesuatu yang harus dievaluasi dan dimodifikasi kembali. Segala perubahan dapat terjadi pada saat prototype dibuat untuk memenuhi kebutuhan pengguna dan pada saat yang sama memungkinkan pengembang untuk lebih memahami kebutuhan pengguna secara lebih baik lagi.

Berikut merupakan manfaat yang diperoleh dari penggunaan *prototyping* adalah :

a) Metode *prototype* memungkinkan pengguna untuk mendapatkan sistem yang lebih baik.
b) Penggunaan metode *prototype* dengan fleksible dapat menyesuaikan kebutuhan user karena *prototype* menerima masukan pengguna sampai akhir.
c) User dengan perancang memiliki waktu yang lebih banyak dalam interaksi seputar kebutuhan yang diinginkan.

Metode *prototype* ini memiliki beberapa tahapan yang memiliki perannya masing - masing selama proses perancangan perangkat lunak yang bisa dijelaskan masing - masing tahapan tersebut pada penjelasan dibawah ini :

1. *Communication*

   Dimulai dengan tahap *communication*, tahapan ini bertujuan untuk mengidentifikasikan berbagai kebutuhan aplikasi yang akan dirancang nantinya dengan melibatkan para client yang bersangkutan agar selama proses perancangan bisa memberikan hasil yang tepat sesuai keinginan *client* yang bersangkutan.

2. *Quick Plan*

   Pada tahap quick plan ini perancang perangkat lunak akan melakukan perencanaan cepat sesuai dengan spesifikasi kebutuhan *user* berdasarkan data yang telah dikumpulkan pada tahap *communication* dengan merancang desain antarmuka yang dibutuhkan dan kebutuhan pendukung pada proses ini.

3. *Modeling Quick Design*

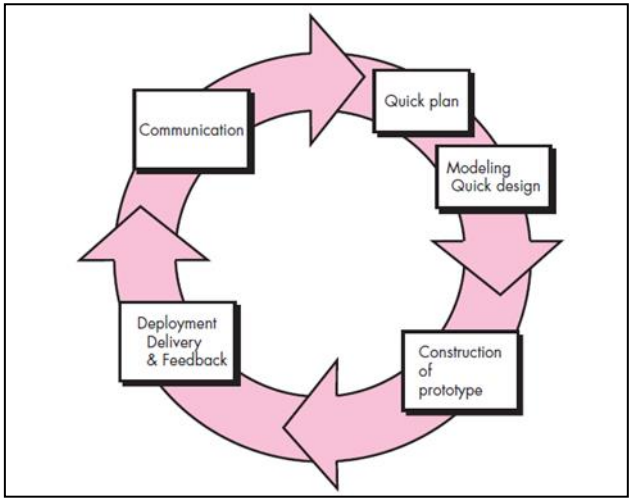
   Pada tahap ini tim perancang akan membuat model design UML ataupun pemodelan yang dibutuhkan lainnya dengan waktu perancangan yang efektif untuk mendeskripsikan kebutuhan *client* berdasarkan analisis yang telah dilakukan sebelumnya.

4. *Construction of Prototype*

Selanjutnya pada tahap ini perancang akan memulai membangunan perangkat lunak berdasarkan data yang telah dikumpulkan sebelumnya, proses pembangunan ini lebih berfokus terhadap aspek utama perangkat lunak dengan maksud pada proses selanjutnya perancang bisa dengan cepat mendapatkan *feedback* dari *client* tentang perangkat lunak yang dibuat.

5. *Deployment Delivery & Feedback*

Dalam tahap ini prototype akan diserahkan kepada *client* untuk mendapatkan *feedback* dari hasil *prototype* tersebut, *feedback* tersebut akan digunakan sebagai landasan untuk memperbaiki *prototype* agar sesuai dengan spesifikasi kebutuhan *client*.



Gambar 2. 1 Metode Prototype

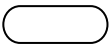
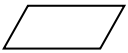
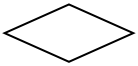
Alat bantu atau *tools-tools* perangkat lunak yang memberikan topangan otomatis ataupun semi-otomatis pada prosesnya dengan metode yang ada seperti *Computer Aided Software Engineering* (CASE).

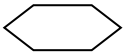
a. Flowchart

Flowchart atau Diagram Alir merupakan bagan (*chart*) yang mengarahkan alir (*flow*) di dalam prosedur atau program sistem secara logika [4]. Flowchart adalah cara untuk menjelaskan tahap-tahap pemecahan masalah dengan merepresentasikan simbol-simbol tertentu yang mudah dipahami, mudah digunakan dan standar. Tujuan penggunaan

flowchart adalah untuk menggambarkan suatu tahapan penyelesaian masalah secara sederhana, terurai dan rapi dengan menggunakan simbol-simbol yang standar yang dapat dimengerti oleh *programmer*.
Berikut ini merupakan simbol-simbol yang terdapat pada *flowchart* seperti pada tabel dibawah ini:

Tabel 2. 1 *Simbol* Flowchart

No	Simbol	Nama	Keterangan
1.		*Terminal*	Memulai dan mengakhiri suatu program
2.		*Input/Output*	Memasukan data maupun menunjukan hasil dari suatu process tanpa tergantung dengan jenis peralatannya
3.		*Flow*	Menghubungkan antara simbol satu dengan simbol yang lain atau menyatakan jalannya arus dalam suatu proses. Simbol arus ini sering disebut juga dengan connecting line.
4.		*Decision*	Memilih proses berdasarkan kondisi yang ada.
5.		*Document*	Merupakan simbol untuk data yang terbentuk informasi.
6.		*Manual Operation*	Menunjukkan pengolahan yang tidak dilakukan oleh komputer/pc.
7.		*Punched Card*	Menyatakan input berasal dari kartu atau output ditulis ke kartu.

No	Simbol	Nama	Keterangan
8.		*Connector*	Menyatakan sambungan dari suatu proses ke proses lainnya dalam halaman / lembar yang sama.
9.		*Offline Connector*	Menyatakan sambungan dari satu proses ke proses lainnya dalam halaman atau lembar yang berbeda.
10.		*Predefined Process*	Menyatakan penyediaan tempat penyimpanan suatu pengolahan untuk memberi harga awal.

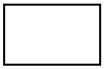
b. UML (*Unified Modeling Language*)

UML *(Unified Modeling Language)* adalah bahasa pemodelan yang digunakan untuk merancang, mendokumentasikan sebuh sistem perangkat lunak, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek.[5] UML muncul dikarenakan adanya kebutuhan pemodelan visual untuk menguraikan, memvisualkan, membentuk dan dokumentasi dari sistem perangkat lunak.

a. Use Case Diagram

Use case diagram atau diagram *use case* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. Secara kasar *use case* digunakan untuk men*get*ahUI fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi–fungsi itu. Berikut Tabel simbol–simbol yang ada pada diagram *use case* :

Tabel 2. 2 Simbol Use Case

No.	Simbol	Nama	Keterangan
1.		*Use case*	Deskripsi dari urutan aksi – aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu *actor*.
2.		*Actor*	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan *use case*.
3.		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
4.		*Include*	Menspesifikasikan bahwa *use case* sumber secara eksplisit.
5.		*Extend*	Menspesifikasikan bahwa *use case* tar*get* memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.
6.		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas

2.1.2 Pemrograman Berorientasi Objek

Pemrograman Berorientasi Objek (OOP)adalah suatucara baru dalam berpikir serta berlogika dalam menghadapi masalah-masalah yang akan dicoba atasi dengan bantuan komputer, dimana setiap objek adalah entitas tunggal yang memiliki kombinasi struktur data dan fungsi tertentu.[6] Pemrograman berorientasi objek berbeda dengan pemrograman prosedural yang hanya menggunakan satu halaman kebawah untuk mengerjakan banyak perintah atau statement. Penggunaan

pemrograman berorientasi objek sangat benyak sekali, contoh : java, php, perl, c#, cobol, dan lainnya.

Objek dalam OOP adalah unit terkecil pemrograman yag masih memiliki data (sifat karakteristik) dan fungsi. Setiap object dapat menerima pesan, memproses data, mengirim, menyimpan, dan memanipulasi data. Class adalah wadah berisi pemodelan suatu objek, mendeskripsikan karakteristik dan fungsi objek tersebut. Karena class merupakan wadah yang akan digunakan untuk menciptakan objek tersebut, maka Class harus diciptakan terlebih dahulu.

OOP memberikan kemudahan dalam pembuatan sebuah program, keuntungan yang didapat apabila membuat program berorientasi objek atau *Object Oriented Programming* (OOP) antara lain :

1) *Reusability*, kode yang dibuat dapat digunakan kembali
2) *Extensibility* , pemrogram dapat membuat methode baru atau mengubah yang sudah ada sesuai yang diinginkan tanpa harus membuat kode dari awal
3) *Maintainability*, kode yang sudah dibuat lebih mudah untuk dikelola apabila aplikasi yang dibuat berskala besar yang memungkinkan adanya errordalam pengembangannya hal tersebut dapat diatasi dengan OOP karena pemrograman OOP sudah menggunakan konsep modularitas.

Terdapat beberapa cara untuk menentukan karakteristik dalam pendekatak berorientasi objek, tetapi secara umum mencakup empat hal, yaitu identifikasi, klasifikasi, polymorphism (polimorfisme) dan inheritance (pewarisan). Metodologi pengembanagn sistem berorientasu objek mempunyai tiga karakteristik utama, yaitu :

1) Encapsulation
Encapsulation (pengkapsulan) merupakan dasar untuk pembahasan ruang lingkup program terhadap data yang diproses.

2) Inheritance
Inheritance (Pewarisan) adalah tehnik yang menyatakan bahwa anak dari objek akan mewarisi data / atribut dan metode dari induknya langsung. Sifat yang dimiliki oleh kelas induknya tidak perlu diulang dalam setiap subkelasnya.

3) Polymorphism
Polymorphism (polimorfisme) yaitu konsep yang menyatakan bahwa sesuatu yang sama dapat mempunyai bentuk dan perilaku berbeda.

2.1.3 Basis Data

Basis data adalah kumpulan data terkait yang disimpan bersama dengan pengurangan redundansi yang terkontrol, dirancang untuk melayani aplikasi secara efisien. Data tersusun sedemikian rupa sehingga tidak bergantung pada program tertentu. Akses untuk menambah, mengedit, dan mengambil data diatur secara terkontrol. Ini juga merupakan sistem penyimpanan file elektronik yang terstruktur.[7] Oleh sebab itu, untuk merancang tabel-tabel yang akan dibuat maka diperlukan pola pikir penyimpanan data nantinya jika dalam bentuk baris-baris data (record) dimana setiap baris terdari beberapa kolom.

Kegunaan utama sistem basis data adalah agar pengguna mampu menyusun suatu pandangan (view) abstraksi data. Hal ini bertujuan untuk menyederhanakan intraksi antara pengguna dengan sistemnya dan basis data dapat mempresentasikan pandangan yang berbeda kepada para pengguna, programmer, dan administratornya. Sebab tidak semua pengguna basis data terlatih dengan baik dan penggunanya terbagi dalam berbagai tingkatan, maka kompleksitas basis data akan tersembbunyi dari para pengguna melalui beberapa level abstraksi data. Ketika memandang basis data, pemakai dapat dikelompokkan menjadi 3 tingkatan yaitu :

1. Level Fisik (physical view/internal view).

   Merupakan tingkatan terendah dalam abstraksi data yang menunjukkan bagaimana data disimpan dalam kondisi sebenarnya. Level ini merupakan bentuk paling kompleks, dimana struktur data level terendah digambarkan pada level ini.

2. Level Konseptual.

   Merupakan level yang menggambarkan data apa yang sebenarnya (secara fungsional) disimpan dalam basis data, beserta relasi yang terjadi antara data. Level ini menggambarkan keseluruhan database, dimana administrator basis data (DBA) membangun dan mengolah basis data, sedangkan pemakai tidak memperdulikan kerumitan dalam struktur level fisik lagi. Contohnya: pengguna akan mengetahui bahwa penjualan disimpan didalam tabel barang, produksi, keuangan, marketing.

3. Level Pandangan Pemakai.

   Merupakan level dengan tingkatan tertinggi, yang menggambarkan hanya satu bagian dari keseluruhan database. Beberapa pengguna basis data tidak membutuhkan semua isi basis data misalkan bagian personalia

hanya membutuhkan data file karyawan dan gaji, tidak membutuhkan data file gudang, transaksi barang masuk.

MySQL merupakan software database open source yang paling populer di dunia. MySQL menjadi pilihan utama bagi banyak pengembang software dan aplikasi hal ini dikarenakan kelebihan MySQL, diantaranya sintaksnya yang mudah dipahami, didukung program-program umum seperti C, C++, Java, PHP, Pyton. MySQL sebenarnya merupakan turunan salah satu konsep utama database sejak lama, yaitu SQL (*Structured Query Language*) .

SQL adalah sebuah konsep pengoperasian database terutama untuk pemilihan atau seleksi dan pemasukan data yang memungkinkan pengoperasia n data dikerjakan denga n mudah secara otomatis. Keadaan suatu sistem database (DBMS) dapat diketahui dari kerja optimizernya dalam melakukan proses perintah-perintah SQL yang dibuat oleh user maupun program-program aplikasinya. SQL dibagi menjadi dua bentuk query yaitu :

1. *Data Definition Language* (DDL)

DDL adalah sebuah *Metode Query* SQL yang berguna untuk mendefinisikan data sebuah *database*, adapun *Query* yang dimiliki adalah:

a) Create : Digunakan untuk pembuatan table dan *database*.
b) Drop : Digunakan untuk penghapusan table dan *database*.
c) Alter : Digunakan untuk pengubahan struktur table yang dibuat, baik menambah *field* (add), mengganti nama *field* (*change*), ataupun menamakannya kembali (*rename*) serta menghapus (drop).

2. *Data Manipulation Language* (DML)

DML adalah sebuah metode *Query* yang dapat digunakan apabila DDL telah terjadi, sehingga fungsi dari *Query* ini adalah untuk melakukan pemanipulasian database yang telah ada atau telah dibuat sebelumnya. Adapun *Query* yang termasuk didalamnya adalah :

a) Insert : Digunakan untuk penginputan data pada table *database*.
b) Update : Digunakan untuk pengubahan data pada table *database*.
c) Delete : Digunakan untuk penghapusan data pada table *database*.

2.1.4 Laporan Keuangan Laba Rugi

Laporan keuangan laba rugi yaitu suatu laporan yang disusun secara sistematis berdasarkan standar akuntansi yang memuat tentang hasil operasi selama atau tahun atau periode akuntansi. Laporan ini menunjukkan sumber dari mana penghasilan diperolah serta beban yang dikeluarkan sebagai beban perusahaan, secara sistematis merupakan laporan tentang penghasilan, beban-beban, dan laba atau rugi.[8]

2.1.5 Metode Single Step

Laporan laba rugi bentuk Single Step sering disebut laporan langsung. Laporan laba rugi ini menggabungkan seluruh pendapatan dan beban perusahaan menjadi satu kelompok, baik pendapatan dan beban operasional maupun non–operasional. Metode Single Step (langsung) adalah seluruh pendapatan yang ada dikelompokkan tersendiri di bagian atas dan dijumlahkan, lalu seluruh beban dikelompokkan tersendiri pula di bagian bawah dan dijumlahkan. Kemudian, jumlah pendapatan dikurangi jumlah beban, lalu selisihnya didapati laba bersih atau rugi bersih.[9]

Pendapatan usaha dihitung berdasarkan total dasar pengenaan pajak dari kegiatan konstruksi dan pengadaan. Dari total pendapatan usaha tersebut, 70% digunakan untuk pembelian bahan dan 17% dialokasikan untuk upah tenaga kerja. Selain itu, biaya BBM dan biaya administrasi proyek diambil dari data yang sudah ditambahkan pada pengeluaran. Total harga pokok penjualan merupakan penjumlahan dari pembelian bahan, upah tenaga kerja, biaya BBM, dan biaya administrasi proyek. Laba bruto diperoleh dengan mengurangkan total harga pokok penjualan dari pendapatan usaha.

Selanjutnya, biaya administrasi dan umum meliputi gaji pegawai tetap, biaya BBM transport, biaya penyusutan, dan biaya administrasi kantor, yang semuanya berasal dari data yang sudah ditambahkan pada pengeluaran. Total biaya administrasi dan umum merupakan hasil penjumlahan dari gaji pegawai tetap, biaya BBM transport, biaya penyusutan, dan biaya administrasi kantor. Laba sebelum pajak dihitung dengan mengurangkan total biaya administrasi dan umum dari laba bruto. Pajak Penghasilan (PPh) final dihitung dengan formula 0.5% dari total pengadaan ditambah 1.75% dari total konstruksi ditambah 2% dari maksimum total pengadaan. Akhirnya, laba bersih diperoleh dengan mengurangkan PPh final dari laba sebelum pajak.

2.1.6 Landasan Hukum Laporan Laba Rugi

Peraturan Pemerintah Republik Indonesia Nomor 64 Tahun 1999 Tentang Perubahan Peraturan Pemerintah No. 24 Tahun 1998 Tentang Informasi Keuangan Tahunan Perusahaan:

Mengubah beberapa ketentuan Peraturan Pemerintah Nomor 24 Tahun

1998 tentang Informasi Keuangan Tahunan Perusahaan sebagai berikut:

(1) Laporan Keuangan Tahunan sebagaimana dimaksud dalam pasal 2 ayat meliputi:

a) Neraca;

b) Laporan laba rugi;

c) Laporan perubahan ekuitas;

d) Laporan arus kas, dan

e) Catatan atas laporan keuangan yang mengungkapkan utang piutang termasuk kredit bank dan daftar penyertaan modal.

(2) Uraian dan rincian Laporan Keuangan Tahunan sebagaimana dimaksud dalam ayat (1) ditetapkan lebih lanjut oleh Menteri setelah mendapat pertimbangan dari Menteri Keuangan."