BAB II
DASAR TEORI

BAB II
DASAR TEORI

2.1 Rekayasa Perangkat Lunak

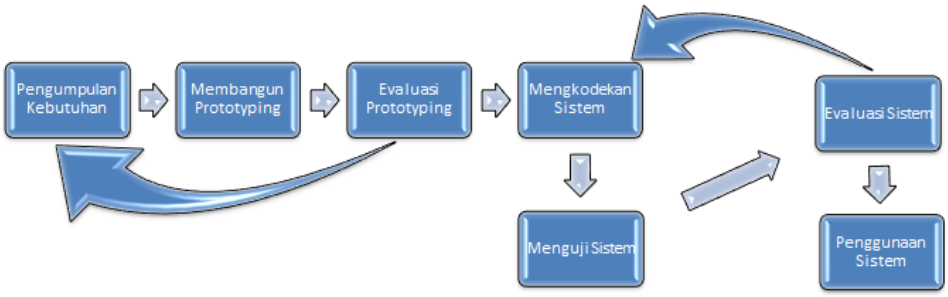
Rekayasa Perangkat Lunak *(software)* adalah seluruh perintah yang digunakan dalam memproses informasi, yang dapat berupa program atau prosedur. Program merupakan kumpulan perintah yang dapat dimengerti oleh komputer, sedangkan prosedur merupakan perintah yang diperlukan oleh pengguna dalam memproses informasi[4].

Rekayasa Perangkat Lunak adalah suatu disiplin ilmu yang membahas seluruh aspek produksi perangkat lunak, dari analisis kebutuhan pengguna, menentukan spesifikasi kebutuhan pengguna, desain, pengkodean, testing hingga pemeliharaan sistem setelah digunakan. Dalam pembuatan perangkat lunak, diperlukan suatu metode pengembangan sistem. Salah satunya adalah menggunakan metode *prototype*.

Metode *Prototype* adalah suatu model pengembangan perangkat lunak yang memungkinkan interaksi yang tinggi antara pengembang sistem dan pengguna sistem, dengan tujuan untuk mengatasi ketidakcocokkan antara pengembang dan pengguna. Menurut Sutoyo metode *prototype* memungkinkan pengguna sistem untuk melihat dan menguji *prototype* dari sistem, yang memberikan umpan balik dan spesifikasi dari sistem yang lebih lengkap dari sistem tersebut.

2.1.1 Metode Pengembangan Sistem

Metode pengembangan sistem yang akan digunakan pada aplikasi *tracer study* berbasis *website* yaitu metode pengembangan perangkat lunak *Software Development Life* (SDLC) model *prototype* (Pressman, 2012). Metode ini membantu pengembang dalam berkomunikasi dengan pengguna supaya mendapatkan hasil yang sesuai dengan keinginan pengguna melalui rancangan tampilan fungsionalitas aplikasi. Berikut ini adalah tahapan metode *prototype* menurut Pressman (2012)[5]. Dalam Gambar 2.1.



Gambar 2. 1 Metode *Prototype* [5]

Berdasarkan model *prototype* yang telah digambarkan di atas, dapat diuraikan pembahasan disetiap tahapannya sebagai berikut[5]:

1. Pengumpulan Kebutuhan

Pengumpulan kebutuhan merupakan tahap awal pada proyek ini yaitu dengan melakukan identifikasi kebutuhan dan membuat garis besar mengenai suatu hal yang diinginkan oleh pengguna terhadap sistem tersebut. Pada tahap ini pengembang berkoordinasi dengan pihak BKK SMK YPE Kroya untuk mengumpulkan kebutuhan data awal dengan cara melakukan observasi dan wawancara secara langsung dengan pengurus BKK SMK YPE Kroya untuk memperoleh data mengenai kebutuhan dan garis besar sistem yang akan dibangun.

2. Perancangan *Prototyping*

Setelah mengetahui dan memahami kebutuhan, tahap berikutnya adalah merancang *prototype*. Pada tahap pembuatan prototype, dilakukan perancangan menggunakan *Unified Modeling Language* (UML), yang meliputi pembuatan *use case diagram*, *entity relationship diagram*, dan *flowchart* sistem yang akan dibangun.

3. Evaluasi *Prototyping*

Pada tahap ini terdapat keterlibatan dengan pengguna untuk mengevaluasi apakah *prototype* yang telah dibangun sesuai dengan kebutuhan dan keinginan pengguna. Jika *prototype* belum sesuai maka mengulang langkah 1 dan 2 untuk melakukan perubahan agar sesuai dengan kebutuhan pengguna. Jika sudah sesuai maka langsung ke tahap berikutnya yaitu pengkodean sistem.

4. Pengkodean Sistem

Pada tahap ini, *prototype* yang telah disetujui akan diterjemahkan dalam bahasa pemograman yang sesuai. Dalam hal ini, bahasa pemograman yang akan digunakan adalah PHP dan *database* MySQL.

5. Pengujian Sistem

Setelah sudah menjadi suatu software yang siap digunakan, harus dilakukan uji coba terlebih dahulu untuk memastikan fungsinya dan mengidentifikasi potensi kesalahan yang ada pada aplikasi, memastikan bahwa dari setiap fitur yang akan digunakan berfungsi dengan benar dan menghasilkan output yang diinginkan. Pengujian ini dilakukan menggunakan metode blackbox testing.

6. Evaluasi Sistem

Pada tahap ini, pengguna melakukan evaluasi terhadap sistem yang telah selesai apakah sesuai dengan yang diharapkan. Jika dinilai sudah sesuai, maka akan ke tahap selanjunya yaitu penggunaan sistem.

7. Menggunakan Sistem (implementasi)

Tahap ini, perangkat lunak *(software)* yang telah melalui uji coba dan diterima oleh pengguna siap untuk digunakan.

2.1.2 Metode Pengujian Sistem

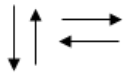
Salah satu metode pengujian sistem pada rekayasa perangkat lunak adalah metode pengujian sistem *Black Box Testing*. Pengujian pada Aplikasi *Tracer Study* Berbasis *Website* ini menggunakan metode *Black Box Testing*. *Black box testing* adalah metode yang hanya berfokus untuk mengamati *input* dan *output* dari sistem dan menjelaskan fungsi sistem tentang bagaimana cara beroperasinya tanpa mengetahui struktur program atau *internal code* dari sistem tersebut. Metode ini digunakan untuk mengetahui apakah perangkat lunak sudah dapat berfungsi dengan benar. Pengujian *black box testing* memiliki tujuan untuk menunjukan fungsi dari sistem tentang cara bekerja, apakah input dan output sudah benar dan sesuai dengan yang diharapkan.

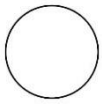
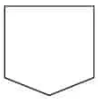
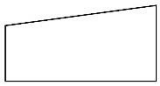
Adapun struktur data merupakan cara menyimpan atau mempresentasikan data didalam komputer agar bisa dipakai secara efisien. Berikut bagian-bagian yang ada pada struktur data:

1. FlowChart

FlowChart berfungsi untuk menggambarkan alur kerja sebuah sistem atau apa yang sedang dikerjakan oleh sebuah sistem secara keseluruhan[6]. *Flowchart* digambarkan dengan simbol-simbol yang mewakili proses tertentu, simbol tersebut kemudian dihubungkan menggunakan garis penghubung untuk membentuk suatu proses atau langkah-langkah sistem. *Flowchart* menjelaskan langkah-langkah yang harus dilakukan dalam sebuah proses, beserta kondisi-kondisi yang harus dipenuhi dan keputusan-keputusan yang harus diambil. Simbol-simbol *flowchart* dapat dilihat pada Tabel 2.1.

Tabel 2. 1 Simbol *FlowChart*

No	Simbol	Nama	Keterangan
1.		*Flow Direction Symbol* / Arus	Untuk menghubungkan antara simbol yang satu dengan simbol yang lain yang menujukan alur sebuah proses. Simbol ini disebut juga *connecting line*.
2.		*Terminator symbol*	Simbol untuk menunjukan mulai (*start*) atau berakhirnya (*end*) dari suatu program.
3.		*Processing Symbol*	Menunjukkan kegiatan atau proses pengolahan yang dilakukan oleh komputer.
4.		*Manual Operation Symbol*	Menunjukkan pengolahan data (proses) yang tidak dilakukan oleh komputer (manual).
5.		*Decision Symbol*	Menunjukan suatu kondisi tertentu yang akan menghasilkan dua kemungkinan (ya/tidak).

6.		*Input-Output*	Menyatakan proses *input* dan *output* tanpa tergantung dari jenis peralatannya.
7.		*Document Symbol*	Menyatakan input berasal dari dokumen dalam bentuk kertas atau output dicetak ke kertas.
8.		*Connector (on-page)*	Menyatakan sambungan dari suatu proses ke proses lainnya dalam halaman/lembar yang sama
9.		*Connector (off-page)*	Menyatakan sambungan dari suatu proses ke proses lainnya dalam halaman/lembar yang berbeda.
10.		*Manual Input*	Memasukan data secara manual dengan menggunakan online keyboard.
11.		*Predefined Proses*	Menyatakan penyediaan tempat penyimpanan suatu pengolahan untuk memberi harga awal.

2. Use Case Diagram

Use Case Diagram adalah pemodelan untuk kelakuan atau *behavior* sebuah sistem informasi yang akan dibuat. *Use Case diagram* berfungsi untuk menjelaskan interaksi antara satu atau lebih *actor* dengan sistem yang akan dibangun dan dapat mendeskripsikan fungsi apa yang ada dalam sebuah sistem informasi. Berikut adalah simbol-simbol dari *use case diagram* seperti yang terlihat pada Tabel 2.2

Tabel 2. 2 Simbol *Use Case Diagram*

No.	Simbol	Nama	Keterangan
1.		*Actor*	Menggambarkan *actor* yang berinteraksi dengan sistem dan dapat menerima dan memberi informasi pada sistem. *Actor* bisa berupa orang, perangkat keras atau objek lain dalam sistem yang sama.
2.		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*independent*) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*not independent*)
3.		*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu *actor*.

4.		*Generalization*	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada diatasnya objek induk (*ancestor*).
5.		*Include*	Menspesifikasikan bahwa *use case* satu merupakan bagian dari *use case* lain.
6.		*Extend*	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.
7.		*Association/* Asosiasi	Menggambarkan navigasi antar *class*, berupa banyak objek lain yang berhubungan dengan satu objek, dan apakah suatu *class* menjadi bagian dari *class* lainnya.
8.		*System Boundary*	Berisi nama dari sistem yang diletakkan di dalam bagian atas *boundary*. Mewakili ruang lingkup sistem atau batasan sistem. *Actor* berada di luar ruang lingkup sistem.

2.2 Pemograman Berorientasi Objek (PBO)

Pemograman berorientasi objek adalah sebuah pendekatan dalam pembangunan perangkat lunak yang terdiri dari sekelompok objek yang berisi data dan operasi yang diberlakukan terhadapnya. Metodologi berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis[7]. Metodologi pengembangan sistem yang berorientasi objek melibatkan beberapa konsep dasar yang perlu dipahami, antara lain:

a. Kelas (*Class*)

Kelas adalah kumpulan objek dengan karakteristik yang sama. Sebuah kelas memiliki atribut, operasi/*method*, relationship dan arti. Suatu kelas dapat diturunkan ke kelas yang lain, dimana atribut dan kelas semula dapat diwariskan ke kelas yang baru.

b.Objek (*Object*)

Objek merupakan suatu entitas yang mampu menyimpan informasi (status) dan mempunyai operasi (kelakuan) yang dapat diterapkan pada status objeknya. Objek mempunyai siklus hidup yaitu diciptakan, dimanipulasi dan dimusnahkan.

c. Metode (*Method*)

Operasi atau metode pada sebuah kelas hampir sama dengan fungsi atau prosedur pada metodologi struktural. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi. Metode atau operasi berfungsi untuk memanipulasi objek itu sendiri.

d. Atribut (*Attribute*)

Atribut dari sebuah kelas adalah variabel global yang dimiliki sebuah kelas. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas objek. Atribut dipunyai secara 14 individual oleh sebuah objek, misalnya berat, jenis, nama, dan sebagainya. Atribut sebaiknya bersifat privat untuk menjaga enkasulapsi.

e. Enkapsulasi (*Encapsulation*)

Pembungkusan atribut data dan layanan (operasi - operasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerjanya.

f. Pewarisan (*Inheritance*)

Mekanisme yang memungkin suatu objek mewarisi sebagian atau seluruh defenisi dan objek lain sebagai bagian dari dirinya.

g. Antarmuka (*Interface*)

Antarmuka atau interface biasa digunakan agar kelas lain tidak dapat mengakses langsung ke suatu kelas, melainkan hanya mengangkses antarmukanya.

h. *Reusabilily*

Pemanfaatan kembali objek yang sudah didefenisikan untuk suatu permasalahan pada permasalahan lainnya yang melibatkan objek tersebut. Misalkan pada sebuah aplikasi peminjaman buku diperlukan pada kelas anggota, maka ketika membuat aplikasi VCD, kelas anggota ini bisa digunakan kembali dengan sedikit perubahan untuk aplikasi penyewaan VCD tanpa harus membuat dari awal lagi.

i. Komunikasi Antar Objek

Komunikasi antara objek dilakukan melalui pesan (*message*) yang dikirim dari satu objek ke objek yang lainnya.

j. Polimorpisme (*Polymorphism*)

Kemampuan suatu objek untuk digunakan di banyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.

k. *Package*

Package adalah sebuah kontainer yang dapat digunakan untuk mengelompokan kelas-kelas sehingga memungkinkan beberapa kelas bernama sama disimpan dalam package yang berbeda.

2.3 Database

Menurut Connolly dan Begg *database* merupakan sekumpulan data yang saling terhubung secara logis dan digunakan bersama, dengan deskripsi data yang dirancang untuk memenuhi kebutuhan informasi organisasi. Dari penjelasan tersebut, dapat disimpulkan bahwa *database* adalah suatu sistem penyimpanan data yang terdiri dari sekumpulan data yang saling terkait secara logis, dirancang untuk memenuhi kebutuhan informasi sebuah perusahaan. Pengguna *database* yaitu orang atau program aplikasi. Orang biasanya mengakses *database* melalui terminal dan mengambil data serta informasi dengan menggunakan bahasa *query*. *Query* merupakan permintaan informasi dari *database*, dan bahasa *query* merupakan bahasa khusus yang mudah digunakan oleh pengguna untuk memungkinkan komputer memberikanjawaban atas *query* tersebut[8].

A. *Database Management System* (DBMS)

Database Management System (DBMS) atau merupakan sebuah perangkat lunak *(software)* yang memungkinkan pengguna untuk mendefinisikan, membuat, mengambil data dan mengontrol akses kepada *database*. DBMS adalah sebuah perangkat lunak (*software)* yang mengintegrasikan antara *database* dengan aplikasi program pada pengguna. DBMS menyediakan berbagai fasilitas seperti[9]:

1. *Data Definition Language* (DDL)

DDL merupakan perintah-perintah yang digunakan sebagai pengoperasian skema struktur pada *database*, DDL memperbolehkan pengguna untuk menjelaskan *database*, misalnya merincikan tipe dan batasan data yang akan disimpan dalam *database.* Perintah yang termasuk dalam DDL yaitu:

a. Create

Create adalah perintah untuk membuat termasuk didalamnya untuk membuat *database* dan

table. Contoh perintah, “CREATE *DATABASE* Customer;”, “CREATE TABLE Customer (customer_id char (15), customer_name char(25), customer_addres char(20), constraint PK_customer_id primary key (customer_id));”.

b. Alter

Alter adalah perintah yang berfungsi untuk merubah struktur tabel yang sebelumnya telah dibuat. Hal tersebut mencakup mengubah nama *table*, menambah kolom, menghapus kolom, mengubah kolom, dan memberi atribut pada kolom. Contoh perintah, “ALTER TABLE Customer ADD jenis_member Varchar(200); Alter Table Buku Drop Column Judul_Buku;”.

c. Drop

Drop adalah perintah untuk menghapus *database* maupun *table*. Contoh perintah “DROP TABLE Buku;”.

2. *Data Manipulation Language* (DML)

DML adalah perintah yang digunakan untuk melakukan pengelolaan pada *database* yang telah dibuat. Perintah penting pada DML yaitu:

a. Insert

Insert adalah perintah yang digunakan untuk memasukkan data baru ke dalam *database.* Perintah ini bisa dijalankan ketika *database* dan *table* sudah dibuat, fungsi insert ini untuk menambah data pada table tersebut. Contoh perintah, “INSERT INTO customer (customer_id, customer_name, customer_addres) values('CS001', 'AAN', 'PASURUAN');

b. Update

Update adalah perintah yang berfungsi utuk memperbarui data pada sebuah tabel. Contoh perintah, “Update customer SET customer_name = “SAFAN” WHERE customer_id = CS001;”.

c. Select

Select adalah perintah yang berfungsi untuk menampilkan dan mengambil data yang telah dimasukkan. Contoh perintah, “SELECT * FROM Buku; SELECT id_buku, nama_buku FROM Buku;”.

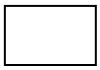
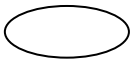
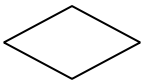
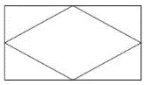
d. Delete

Delete adalah perintah untuk menghapus data. Contoh perintah, “DELETE FROM customer WHERE customer_name = “SAFAN”;”.

3. Entity RelationShip Diagram (ERD)

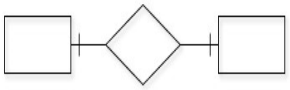
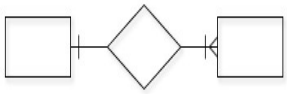
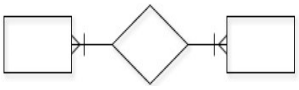
Entity RelationShip Diagram (ERD) merupakan suatu rancangan atau bentuk hubungan suatu kegiatan di dalam sebuah sistem yang berkaitan langsung dan mempunyai fungsi didalam proses tersebut. Pada ERD suatu objek disebut *entity* dan hubungan yang dimilikinya disebut *relationship* atau hubungan antar entitas. Suatu *entity* bersifat unik dan memiliki atribut, hal tersebut dimaksudkan untuk menjadi pembeda dengan *entity* lainnya. Adapun simbol-simbol dari ERD yang dapat dilihat pada Tabel 2.3.

Tabel 2. 3 Simbol *Entity RelationShip Diagram* (ERD)

No.	Nama	Simbol	Keterangan Fungsi
1.	Entitas		Persegi panjang menyatakan himpunan entitas adalah orang, kejadian, atau berada.
2.	Atribut		Atribut merupakan informasi yang diambil tentang sebuah entitas.
3.	Relasi		Belah ketupat menyatakan himpunan relasi merupakan hubungan antar entitas. Relasi biasanya diawali dengan kata kerja.
4.	Link		Garis sebagai penghubung antara himpunan, relasi, dan himpunan entitas dengan atributnya
5.	Associative Entity		Entitas yang digunakan pada relasi many-to-many.

ERD memiliki derajat relasi atau biasa disebut dengan kardinalitas. Kardinalitas mendeskripsikan batasan jumlah keterhubungan suatu *entity* dengan *entity* lainnya. Berikut merupakan macam-macam kardinalitas yang dapat dilihat pada Tabel 2.4.

Tabel 2. 4 Macam-Macam Kardinalitas

No.	Simbol	Nama	Keterangan
1.		Relasi Satu ke Satu (*One to One*)	Relasi yang menunjukkan bahwa setiap himpunan entitas berhubungan dengan tepat satu himpunan entitas lainnya
2.		Relasi Satu ke Banyak (*One to Many*)	Relasi yang menunjukkan bahwa hubungan antara entitas pertama dengan entitas kedua adalah satu banding banyak, begitu pula sebaliknya
3.		Relasi Banyak ke Banyak (*Many to Many*)	Relasi yang menunjukkan bahwa setiap himpunan entitas boleh berhubungan dengan banyak himpunan entitas lainnya dan sebaliknya

2.4 Pendidikan

Bapak Pendidikan Nasional Indonesia, Ki Hajar Dewantara, mendefinisikan bahwa "Pendidikan merupakan suatu kebutuhan atau tuntutan dalam hidup tumbuhnya anak-anak,

dimana tujuannya adalah untuk membimbing segala potensi alamiah yang dimiliki oleh anak-anak tersebut, sehingga mereka sebagai individu dan sebagai anggota masyarakat, mampu mencapai tingkat keselamatan dan kebahagiaan yang setinggi-tingginya"[1]. Pendidikan meliputi segala faktor yang mempengaruhi pertumbuhan, perubahan, dan keadaan setiap individu. Perubahan yang terjadi yaitu dalam hal mengembangkan potensi anak didik dalam berbagai aspek kehidupan, termasuk pengetahuan, keterampilan, dan sikap[1]. Pendidikan tidak terlepas dari seorang alumni.

Definisi alumni menurut Kamus Besar Bahasa Indonesia (KBBI) adalah individu yang telah menyelesaikan atau lulus dari sebuah sekolah atau perguruan tinggi. Alumni merupakan elemen penting yang tidak terlepas dari siklus Pendidikan. Mereka bertindak sebagai perantara antara institusi Pendidikan dengan dunia global[8]. Data yang digunakan sebagai perantara tersebut dapat didapatkan melalui pelacakan alumni *(Tracer study).*

Tracer Study atau survei alumni merupakan sebuah penelitian yang dilakukan untuk mengkaji keadaan alumni dari lembaga pendidikan. Tracer study ini bermanfaat untuk menilai sejauh mana alumni mampu memberikan kontribusi yang relevan dalam pembangunan sesuai dengan latar belakang pendidikannya. *Tracer study* adalah sebuah pendekatan yang digunakan oleh sekolah untuk memperoleh informasi tentang keadaan alumni terutama dalam hal pencarian kerja, situasi saat ini, dan penggunaan kompetensi yang diperoleh selama berada di lembaga pendidikan[2].

Dalam dunia pendidikan terdapat Rekayasa Perangkat Lunak (RPL) yang dapat diimplementasikan salah satunya dalam pembuatan Aplikasi *Tracer Study* Berbasis *Website.*

(~~*Halaman ini sengaja dikosongkan*~~)