BAB II
LANDASAN TEORI

BAB II
LANDASAN TEORI

2.1 Sistem Informasi

Sistem informasi adalah cara-cara yang di organisasi untuk mengumpulkan, memasukan, mengolah, dan menyimpan data, serta untuk menyimpan, mengelola, mengendalikan dan melaporkan informasi sedemikian rupa sehingga sebuah organisasi dapat mencapai tujuan yang ditetapkan.

Menurut Frisdayanti, sistem informasi adalah serangkaian prosedur formal dimana data dikumpulkan. Dalam sistem informasi juga ada input, model, proses, output, penyimpanan dan *control*, sehingga sistem informasi dapat digunakan untuk merencanakan, mengolah, mengendalikan serta meracik data dalam suatu organisasi [2].

Sehingga dapat disimpulkan bahwa sistem informasi adalah kumpulan prosedur resmi yang diatur untuk mengumpulkan, memasukkan, mengolah, menyimpan, mengelola, mengendalikan, dan melaporkan data, sehingga organisasi dapat merencanakan, mengolah, mengendalikan, dan mencapai tujuan yang diinginkan.

2.2 Pendidikan

Pendidikan menjadi sebuah langkah awal dalam seseorang meningkatkan mutu kehidupan. Melalui pendidikan seseorang dapat membuka peluang dan memperluas kesempatan untuk mencapai tujuan dan cita-cita hidup. Sehingga ditarik kesimpulan bahwa Pendidikan adalah segala sesuatu yang menyangkut proses perkembangan manusia yang diupayakan sekolah terhadap anak dan remaja agar mempunyai kemampuan yang sempurna dan kesadaran penuh terhadap hubungan – hubungan dan tugas – tugas sosial mereka [1].

Sekolah merupakan sebuah lembaga formal yang didirikan dengan tujuan memberikan Pendidikan dan berperan sebagai sarana sosialisasi, pembentukan karakter serta keterampilan bagi peserta didiknya. Proses perkembangan dan pengembangan manusia tidak lepas dari proses pembelajaran yang dilakukan oleh seorang pendidik yakni guru kepada pendidik. Tugas dari seorang guru selain mendidik adalah membentuk karakter, membimbing dan memberikan motivasi. Selain tugas pokok sebagai tenaga pendidik, guru juga mendapat tugas tambahan

tergantung pada kebutuhan dan situasi di sekolah seperti contoh menjadi guru piket, pembina, wakil kepala (WaKa) kepala satuan,dan lainnya.

Guru piket di SMA Negeri 1 Cilacap mempunyai tugas diantaranya mencatat ketidakhadiran siswa di masing-masing kelas. Menyampaikan tugas yang dititipkan oleh guru mata pelajaran yang berhalangan hadir dikarenakan Tugas Dinas, Cuti, atau Ijin serta mendata guru yang berhalangan hadir tersebut. Lalu kemudian menuliskannya di Buku Daftar Pemantauan Absensi, Laporan Guru Jaga dan Laporan Pendistribusian Tugas.

2.3 Rekayasa Perangkat Lunak

Perangkat Lunak (*software*) adalah program komputer yang terasosiasi dengan dokumentasi perangkat lunak seperti dokumentasi kebutuhan, model desain, dan cara penggunaan (*user* manual) [3]. Dalam perangkat lunak tidak hanya berisi program, tetapi juga semua dokumentasi dan konfigurasi data yang paling 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-teknik seperti ketidakmauan pengguna untuk mengubah cara kerja dari manual ke otomatis, atau ketidakmampuan pengguna menggunakan computer. 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 Pressman, dalam melakukan perancangan sistem yang akan dikembangkan dapat menggunakan *prototype* [4]. 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 guru piket. 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 fleksibel 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 di bawah 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.

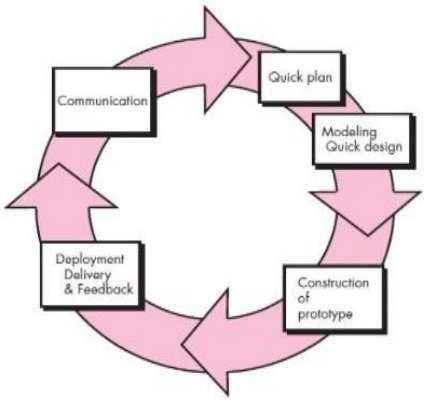
4) Construction of Prototype

Selanjutnya pada tahap ini perancang akan memulai membangun 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*. Berikut ini merupakan gambar metode *Prototype* yang tertera pada Gambar 2.1.



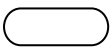
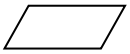
Gambar 2. 1 Metode *Prototype*[5]

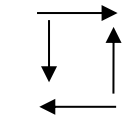
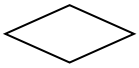
Alat bantu atau *tools-tools* perangkat lunak yang memberikan topangan otomatis ataupun semi-otomatis pada prosesnya dengan metode yang ada seperti *Computer Aided Software Engineering* (CASE).

a. Flowchart

Flowchart atau Diagram Alir merupakan bagan (*chart*) yang mengarahkan alir (*flow*) di dalam prosedur atau program sistem secara logika [6]. Flowchart adalah cara untuk menjelaskan tahap – tahap pemecahan masalah dengan merepresentasikan simbol-simbol tertentu yang mudah dipahami, mudah digunakan dan standar[7]. Tujuan penggunaan *flowchart* adalah untuk menggambarkan suatu tahapan penyelesaian masalah secara sederhana, terurai dan rapi dengan menggunakan symbol-simbol yang standar yang dapat dimengerti oleh programmer. Berikut ini merupakan simbol-simbol yang terdapat pada *flowchart* seperti pada Tabel 2.1 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.		*Process*	Digunakan untuk menunjukan pengolahan yang akan dilakukan dalam computer.

b. UML (*Unified Modeling Language*)

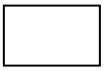
UML (*Unified Modeling Language*) adalah salah standar Bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek [3]. UML muncul dikarenakan adanya kebutuhan pemodelan visual untuk menguraikan, memvisualisasikan, 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 mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu. Berikut ini merupakan simbol – simbol yang ada pada diagram *use case* seperti pada Tabel 2.2 dibawah ini:

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*.

No.	Simbol	Nama	Keterangan
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.
6.		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas

2.4 Pemrograman Berorientasi Objek

Pemrograman berorientasi objek atau (*Object Oriented Programming*/OOP) merupakan suatu pendekatan pemrograman yang semua data dan fungsinya dibungkus menggunakan *object* dan *class* [8]. Pemrograman berorientasi objek berbeda dengan pemrograman prosedural yang hanya menggunakan satu halaman ke bawah untuk mengerjakan banyak perintah atau statement. Penggunaan pemrograman berorientasi objek sangat banyak sekali, contoh : java, php, perl, c#, cobol, dan lainnya.

Objek dalam OOP adalah unit terkecil pemrograman yang masih memiliki data (sifat karakteristik) dan fungsi. Setiap objek 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*, pemrograman dapat membuat metode 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 error dalam pengembangannya hal tersebut dapat diatasi dengan OOP karena pemrograman OOP sudah menggunakan konsep modularitas.

Terdapat beberapa cara untuk menentukan karakteristik dalam pendekatan berorientasi objek, tetapi secara umum mencakup empat hal, yaitu identifikasi, klasifikasi, *polymorphism* (polimorfisme) dam *inheritance* (pewarisan). Metodologi pengembangan sistem berorientasi 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 Teknik 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.

2.5 Basis data

Sistem basis data adalah sistem terkomputerisasi yang tujuan utamanya untuk memelihara data yang sudah diolah dan membuat informasi tersedia saat dibutuhkan [9]. Basis data adalah media untuk menyimpan, mengambil kembali data dan membuat laporan berdasarkan data yang telah disimpan. 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 terdiri dari beberapa kolom.

Kegunaan utama sistem basis data adalah agar pengguna mampu menyusun suatu pandangan (*view*) abstraksi data. Hal ini bertujuan untuk menyederhanakan interaksi antara pengguna dengan sistemnya dan basis data dapat mempresentasikan pandangan yang berbeda kepada para pengguna, programmer, dan administrator nya. Sebab tidak semua 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 menunjukan 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 penggunaan 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.

SQL adalah sebuah konsep pengoperasian database terutama untuk pemilihan atau seleksi dan pemasukan data yang memungkinkan pengoperasian data dikerjakan dengan 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 tabel dan *database*.
   b) *Drop* : Digunakan untuk penghapusan tabel dan *database*.
   c) *Alter* : Digunakan untuk pengubahan struktur tabel yang dibuat, baik menambah *field* (*add*), mengganti nama *field* (*change*), ataupun menamakannya kembali (*rename*) serta menghapus (*drop*)

2. *Data Manipulation Language* (DML)

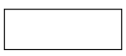
   DML adalah sebuah metode *Query* yang dapat digunakan apabila DDL telah terjadi, sehingga fungsi dari *Query* ini adalah untuk melakukan pemanipulasian database yang telah ada atau telah dibuat sebelumnya. Adapun *Query* yang termasuk didalamnya adalah :

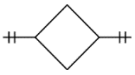
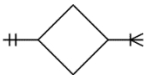
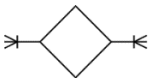
a) *Insert* : Digunakan untuk penginputan data pada tabel *database*.
b) *Update* : Digunakan untuk pengubahan data pada tabel *database*.
c) *Delete* : Digunakan untuk penghapusan data pada tabel *database*.

ERD (*Entity Relationship Diagram*) adalah model konseptual yang mendeskripsikan hubungan antara penyimpanan. ERD digunakan untuk memodelkan struktur data dan hubungan antar data, dengan menggunakan ERD model dapat diuji dengan mengabaikan proses yang dilakukan. ERD pertama kali dideskripsikan oleh Peter Chen yang dibuat sebagai bagian dari perangkat lunak CASE. Komponen – komponen yang termasuk dalam ERD antara lain, adalah [10]:

1. Entitas (*Entity*)
   Sebuah produk atau objek yang dapat dibedakan dari obyek lain.
2. Relasi (*Relationship*)
   Asosiasi 2 (dua) atau lebih entitas dan berupa kata kerja.
3. Atribut (*Attribute*)
   Properti yang dimiliki setiap entitas yang akan disimpan datanya.
4. Kardinalitas (*Kardinality*)
   Angka yang menunjukan banyaknya kemunculan suatu objek terkait dengan kemunculan objek lain pada suatu relasi. Kardinalitas relasi yang terjadi diantara dua himpunan entitas (misalnya A dan B) dapat berupa :
   a. Modalitas (*Modality*) adalah partisipasi sebuah entitas pada suatu relasi, 0 jika partisipasi bersifat "*Optional*/parsial", dan 1 jika partisipasi bersifat "wajib/total".
   b. Total *Constraint* adalah *constraint* yang mana data dalam entitas yang memiliki *constraint* tersebut terhubung secara penuh ke dalam entitas dari relasinya. Berikut ini merupakan simbol-simbol yang terdapat pada ERD (*Entity Relationship Diagram*) seperti pada Tabel 2.3 dibawah ini:

Tabel 2. 3 Simbol ERD

No.	Simbol	Nama	Keterangan
1.		Entitas	Individu yang mewakili suatu objek dan dapat dibedakan dengan objek yang lain.

No.	Simbol	Nama	Keterangan
2.		Atribut	Properti yang dimiliki oleh suatu entitas.
3.		Atribut Kunci	Atribut yang dapat mengidentifikasi sebuah entitas secara unik.
4.		Relasi	Menunjukan hubungan di antara sejumlah entitas yang berbeda.
5.		Entitas Asosiatif	Menunjukan hubungan *many-to-many* antara dua entitas atau lebih.
6.		Relasi 1 : 1	Relasi yang menunjukan bahwa setiap entitas pada himpunan entitas pertama berhubungan dengan paling banyak satu entitas pada himpunan entitas kedua.
7.		Relasi 1 : N	Relasi yang menunjukan bahwa hubungan antara entitas pertama dengan entitas kedua adalah satu banding banyak atau sebaliknya. Setiap entitas dapat berelasi dengan banyak entitas pada himpunan entitas yang lain.
8.		Relasi N : N	Hubungan ini menunjukan bahwa setiap entitas pada himpunan entitas yang pertama dapat berhubungan dengan banyak entitas pada himpunan entitas yang kedua, demikian juga sebaliknya.