BAB II

LANDASAN TEORI

2.1 Sistem

Sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran yang tertentu. (Jogiyanto, 2005)

Pendekatan sistem yang merupakan jaringan kerja dari prosedur lebih menekankan urut-urutan operasi di dalam system (Richard F. Neuschel, 2005),” Prosedur adalah sutu urut-urutan operasi klerikal (tulis menulis), biasanya melibatkan beberapa orang didalam satu atau lebih departemen, yang diterapkan untuk menjamin penanganan yang seragam dari transaksi-transaksi bisnis yang terjadi”.

Suatu sistem mempunyai tujuan (*goal*) atau sasaran (objektifitas). Tujuan biasanya dihubungkan dengan ruang lingkup yang lebih luas dan sasaran dalam ruang lingkup yang lebih sempit. Sasaran menentukan masukan dan keluaran yang dihasilkan. Sistem dikatakan berhasil jika mencapai suatu sasaran dan tujuan.

2.2 Aplikasi

Aplikasi adalah perangkat lunak yang ada pada komputer digunakan untuk melayani berbagai macam kebutuhan. Teknologi yang canggih dari perangkat keras akan berfungsi bila instruksi-instruksi tertentu telah diberikan kepadanya. Instruksi-instruksi tersebut disebut dengan perangkat lunak (Jogiyanto, 2005).

Definisi lain aplikasi adalah program yang dibuat oleh pemakai yang ditujukan untuk melakukan suatu tugas khusus. Program seperti ini biasa

dikelompokkan menjadi dua yaitu program aplikasi serbaguna dan program aplikasi spesifik.

Program aplikasi serbaguna adalah program aplikasi yang dapat digunakan oleh pemakai untuk melaksanakan hal-hal yang bersifat umum (misalnya untuk membuat dokumen atau untuk pengiriman surat secara elektronis) serta untuk mengotomasikan tugas-tugas individual yang bersifat berulang (misalnya untuk melakkan perhitungan yang bersifat rutin). Termasuk dalam kategori ini antara lain adalah DBMS sederhana, *Web browser*, surat elektronis, pengolah kata (*word processor*), dan lembar kerja (*spreadsheet*). Program aplikasi serbaguna seringkali disebut perangkat lunak pemakai akhir (*end-user-software*).

Program aplikasi spesifik adalah program yang ditujukan untuk menangani hal-hal yang sangat spesifik. Misalnya, program pada sistem POS (*point-of-sale*) dan ATM. Termasuk dalam kategori ini adalah program yang disebut sebagai paket aplikasi atau perangkat lunak paket (Kadir, 2014).

2.3 Penjadwalan

2.3.1 Pengertian Penjadwalan

Penjadwalalan (*scheduling*) dapat di definisikan sebagai pengalokasian sumber daya dalam waktu tertentu untuk melakukan serangkaian tugas (Baker, 1974). Menurut Morton (1993), penjadwalan adalah proses pengorganisasian, pemilihan, dan penentuan waktu penggunaan sumber-sumber untuk mengerjakan semua aktifitas yang diperlukan yang memenuhi kendala aktifitas dan sumber daya.

Penjadwalan merupakan suatu kegiatan penting dalam perusahaan. Dalam suatu lembaga pendidikan, penjadwalan dilakukan untuk mealokasikan ruang

kelas, peralatan mengajar, tenaga mengajar, *staff* administrasi dan lain sebagainya. Demikian pula kegiatan perhotelan, penjadwalan diperlukan dalam pengaturan kamar hotel, ruang seminar atau resepsi, menu makanan ataupun *entertainer*. Terlepas dari jenis perusahaan, setiap perusahaan perlu untuk melakukan penjadwalan sebaik mungkin agar memperoleh utilitas maksimum dari sumber daya produksi dan aset yang dimiliki.

Melaksanakan pekerjaan secara efektif dan efisien agar tujuan tercapai adalah yang dinginkan oleh semua manajemen perusahaan. Oleh karena itu pemahaman konsep penjadwalan sangat penting, sehingga para pelaksana mengetahui kapan harus memulai suatu pekerjaan dan kapan waktu mengakhirinya.

Penjadwalan adalah proses yang sering kita temui sehari hari. Pengaturan sumber daya yang tersedia sangat terbatas untuk berbagai kegiatan yang membutuhkannya pada periode waktu tertentu ini adalah inti dari masalah penjadwalan. Besarnya ukuran ruang pencarian solusi untuk masalah penjadwalan membuat proses penyusunan jadwal sangat susah dilakukan secara *manual*.

Penjadwalan yang baik akan membuat penggunaan *resource* menjadi lebih efektif. Oleh karena itu, suatu proses kegiatan harus dialokasikan dengan baik dan memenuhi batasan-batasan yang ada. Permasalahan utama yang menjadi kendala dalam melakukan proses penjadwalan adalah data (*resource* dan batasan) yang banyak dan frekuensi perubahan yang tinggi pada data. Dengan permasalahan seperti ini, proses komputasi dibutuhkan untuk memudahkan prses penjadwalan. Agar batasan dalam bahasa manusia dapat dikomputasi dan memberikan hasil sesuai dengan yang diinginkan.

2.3.2 Prinsip Penjadwalan

Prinsip penjadwalan menurut Suryadi H.S (1994) dibagi menjadi 3. Yaitu jadwal tetap, jadwal flexible dan keterbatasan (konflik dan hambatan), berikut ini merupakan penjelasannya:

1. Jadwal Tetap

Jadwal tetap sama dengan jadwal kereta api, dimana setiap aktiftas selalu dijalankan atau terjadi dalam waktu yang sama pada hari yang sama. Jadwal tetap hanya dapat diterapkan apabila setiap pekerjaan menjadi bagian dari *set routine*. Namun selain memperhatikan pekerjaan yang produktif kita juga harus memperhatikan:

1. Pekerjaan khusus yang *on-time* (sekali dikerjakan)
2. Rerun (Perjalanan kembali) pekerjaan produksi karena terjadinya kesalahan atau kegagalan.
3. Pengembangan dan pemeliharaan sistem dan program.
4. Perawatan pekerjaan, seperti pengkopian pita, pengujian, dan sebagaianya.

Untuk sebagian besar instalasi, jadwal tetap sangat terlibat dalam penyusunan atau penetapan waktu *input* serta *output* dan akan terbukti terlalu terbatas.

2. Jadwal Fleksibel

Kebanyakan instalasi memerlukan sistem penjadwalan yang fleksibel untuk penerimaan *input* dan penyampaian *output* sepanjang berbagai tahapan pemrosesan yang didasarkan pada:

1. Waktu yang bisa diterima
2. Waktu optimum
3. *Deadline*

Dalam kegiatan normal, *output* bisa muncul atau didapat dalam waktu optimum jika *input* yang diterima dalam waktu *deadline*. Dimungkinkan pula, meskipun tidak dijamin, hasil yang diproduksi bisa didapat secara *on time* (tepat waktu) meskipun *outputnya* terlambat.

3. Konflik dan Hambatan (Keterbatasan)

Masalah penjadwalan menjadi konflik tetap antara pemenuhan kepada departemen pemakai dengan cara menghasilkan output secara tepat waktu dengan pencapaian keluaran yang maksimum dengan muatan kerja yang seimbang, atau dengan kata lain antara layanan maksimum dengan efektivitas biaya.

Di satu sisi, hambatan atau keterbatasan diakibatkan oleh keperluan aktifitas pemrosesan, dan di sisi lain oleh kemampuan pemakai untuk mengontrol waktu *input* dan *volume*.

2.3.3 Tujuan Penjadwalan

Kegunaan suatu pekerjaan perlu adanya penjadwalan yaitu:

1. Mengurangi timbunan pekerjaan yang terlalu banyak dan kepadatan pemrosesan.
2. Membuat efektif penggunaan sumber daya
3. Memenuhi keperluan pemakai.

Jadwal harus bersifat:

1. Fleksibel untuk dijadwal ulang apabila diperlukan
2. Berisi informasi yang memadai
3. Siap tersedia

Dengan memperhatikan kedua hal tersebut yaitu kegunaan serta sifat penjadwalan, maka tujuan penjadwalan dapat dicapai sehingga pekerjaan dapat diselesaikan secara tepat waktu.

2.4 *Monitoring*

2.4.1 Pengertian *Monitoring*

Monitoring adalah penilaian yang terus menerus terhadap fungsi kegiatan-kegiatan proyek di dalam konteks jadwal-jadwal pelaksanaan dan terhadap penggunaan input-input proyek oleh kelompok sasaran di dalam konteks harapan-harapan rancangan. *Monitoring* adalah kegiatan proyek yang integral, bagian penting dari praktek manajemem yang baik dan karena itu merupakan bagian yang integral dari manajemen sehari-hari (Casely & Kumar 1987).

Monitoring adalah kegiatan mengamati perkembangan pelaksanaan rencana suatu kegiatan, mengidentifikasi serta mengantisipasi permasalahan yang timbul dan atau akan timbul untuk dapat diambil tindakan sedini mungkin.

Monitoring adalah proses pengumpulan dan analisis informasi (berdasarkan indikator yang ditetapkan) secara sistematis dan kontinu tentang kegiatan program / proyek sehingga dapat dilakukan tindakan koreksi untuk penyempurnaan program / proyek itu selanjutnya (Harry Hikmat 2010).

Monitoring adalah proses rutin pengumpulan data dan pengukuran kemajuan atas objektif program. Memantau perubahan, yang *focus* pada proses dan keluaran. *Monitoring* melibatkan perhitungan atas apa yang kita lakukan. *Monitoring* melibatkan pengamatan atas kualitas dari layanan yang kita berikan

2.4.2 Tujuan *Monitoring*

Terdapat 5 tujuan dalam *monitoring*, antara lain:

1. Mengkaji apakah kegiatan-kegiatan yang dilaksanakan telah sesuai dengan rencana.
2. Mengidentifikasi masalah yang timbul agar langsung dapat diatasi
3. Melakukan penilaian apakah pola kerja dan manajemen yang digunakan sudah tepat untuk mencapai tujuan proyek.
4. Mengetahui kaitan antara kegiatan dengan tujuan untuk memperoleh ukuran kemajuan.
5. Menyesuaikan kegiatan dengan lingkungan yang berubah, tanpa menyimpang dari tujuan

2.4.3 Fungsi *Monitoring*

Terdapat 4 fungsi dalam *monitoring*, antara lain:

1. Ketaatan (*compliance*)

Monitoring menentukan apakah tindakan administrator, staf, dan semua yang terlibat mengikuti standar dan prosedur yang telah ditetapkan

2. Pemeriksaan (*auditing*)

Monitoring menetapkan apakah sumber dan layanan yang diperuntukkan bagi pihak tertentu bagi pihak tertentu (target) telah mencapai mereka.

3. Laporan (*accounting*)

Monitoring menghasilkan informasi yang membantu "menghitung" hasil perubahan sosial dan masyarakat sebagai akibat implementasi kebijaksanaan sesudah periode waktu tertentu

4. Penjelasan (*explanation*)

Monitoring menghasilkan informasi yang membantu menjelaskan bagaimana akibat kebijaksanaan dan mengapa antara perencanaan dan pelaksanaannya tidak cocok.

2.4.4 Prinsip-Prinsip *Monitoring*

Terdapat 7 prinsip-prinsip dalam *monitoring*, antara lain:

1. *Monitoring* harus dilakukan secara terus-menerus
2. *Monitoring* harus menjadi umpan terhadap perbaikan kegiatan *program* organisasi
3. *Monitoring* harus memberi manfaat baik terhadap organisasi maupun terhadap pengguna produk atau layanan.
4. *Monitoring* harus dapat memotifasi staf dan sumber daya lainnya untuk berprestasi
5. *Monitoring* harus berorientasi pada peraturan yang berlaku
6. *Monitoring* harus obyektif
7. *Monitoring* harus berorientasi pada tujuan program

2.4.5 Manfaat *Monitoring*

Manfaat *monitoring* bagi penanggung jawab:

1. Salah satu fungsi manajemen yaitu pengendalian atau supervisi. Sebagai bentuk pertanggungjawaban (akuntabilitas) kinerja.
2. Untuk meyakinkan pihak-pihak yang berkepentingan.
3. Membantu penentuan langkah-langkah yang berkaitan dengan kegiatan selanjutnya.
4. Sebagai dasar untuk melakukan *monitoring* dan evaluasi selanjutnya.

Manfaat *monitoring* bagi pengelola:

1. Membantu untuk mempersiapkan laporan dalam waktu yang singkat
2. Mengetahui kekurangan-kekurangan yang perlu diperbaiki dan menjaga kinerja yang sudah baik.
3. Sebagai dasar (informasi) yang penting untuk melakukan evaluasi proyek

2.4.6 Aspek *Monitoring*

Terdapat 3 aspek *monitoring*, antara lain:

1. Aspek masukan (*input*) kegiatan antara lain mencakup: tenaga manusia, dana, bahan, peralatan, jam kerja, data, kebijakan, manajemen dan sebagaimya. Yang dibutuhkan untuk melaksanakan kegiatan kegiatan.
2. Aspek proses / aktivitas yaitu aspek dari kegiatan yang mencerminkan suatu proses kegiatan, seperti penelitian, pelatihan, proses produksi, pemberian bantuan dan sebagainya.
3. Aspek keluaran (*output*), yaitu aspek kegiatan yang mencakup hasil dari proses yang terutama berkaitan dengan kuantitas (jumlah).

2.4.7 Langkah-Langkah Dalam *Monitoring*

Terdapat 3 langkah-langkah dalam *monitoring*, antara lain:

1. Perencanaan
   a) Merancang sistem monitoring yang spesifik: apa yang akan dimonitor, mengapa dan untuk siapa (*user*).
   b) Menentukan *scope monitoring*: luasnya area (Rumah Sakit, puskesmas non TT)? Apakah bersifat klinis atau *service*? Siapa yang terlibat? Berapa lama monitoring akan dilakukan?

c) Memilih dan menentukan indikator menentukan batasan sasaran kelompok misalnya kelompok anak dibawah 2 tahun, 5 tahun atau antara 12-60 bulan

d) Menentukan sumber-sumber informasi, memilih metoda pengumpulan data, seperti metoda observasi, *interview* petugas, rapid *survey* untuk cakupan atau pengobatan di rumah (*home treatment*).

2. Implementasi

a) Memilih dan menentukan proses supervisi dan prosessingnya (kemana akan dikirim).

b) Tabulasi data dan analisa data: membandingkan temuan atau pencapaian aktual dengan perencanaan.

c) Temuan dalam *monitoring*: apakah ada penyimpangan, bila ada perlu diidentifikasi masalah penyebabnya.

d) Menggali penyebab dan mengambil tindakan perbaikan: menggali penyebab terjadinya masalah. Rencana *monitoring* perlu disusun jangka pendek untuk menjamin bahwa tindakan/prosedure dilaksanakan sesuai standar (rencana) serta memberi efek sesuai dengan harapan.

3. Menentukan kelanjutan *monitoring*

Kegiatan *monitoring* dirancang untuk memperoleh hasil kinerja sekarang atau jangka pendek bagi manajer atau user lainnya. Ketika program atau kegiatan rutin telah memberikan perubahan signifikan, maka kelangsungan program kinerja memerlukan perhatian. *Review* secara periodik tetap diperlukan. Sistem informasi manajemen akan membantu manajer untuk mempertimbangkan kapan indikator

dan frekuensi *monitoring* dikurangi dan pada bagian mana perlu direncanakan lagi dan dilanjutkan.

2.4.8 Tipe *Monitoring*

Terdapat 2 tipe dalam *monitoring*, antara lain:

1. *Monitoring* Rutin

Kegiatan mengkompilasi informasi secara reguler berdasarkan sejumlah indikator kunci. *Monitoring* rutin dapat dipergunakan untuk mengidentifikasi penerapan program dengan atau tanpa perencanaan

2. *Monitoring* jangka Pendek

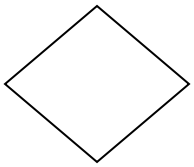
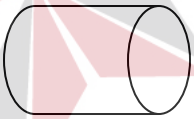
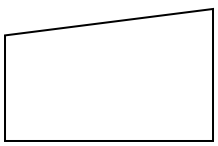
Dilakukan untuk jangka waktu tertentu dan biasanya diperuntukkan bagi aktifitas yang spesifik. Seringkali bila aktifitas atau proses-proses baru diterapkan, manajer ingin mengetahui, apakah sudah diterapkan sesuai rencana dan apakah sesuai dengan keluaran yang diinginkan.

2.5 Bagan Alir Sistem

Menurut Basuki (2003), Sistem flow adalah bagian yang menunjukkan arus pekerjaaan secara menyeluruh dari suatu sistem dimaana bagan ini menjelaskan urutan prosedur-prosedur yang ada dalam sistem dan biasanya dalam membuat sistem flow sebaiknya ditentukan pada fungsi yang melaksanakan atau bertanggung jawab terhadap sub-sub sistem. Bagan alir sistem menggunakan simbol sebagaimana terdapat pada tabel 2.1

Tabel 2.1 Simbol Bagan Aliran Sistem

No	Simbol	Nama Simbol	Keterangan
1		Dokumen	Simbol ini digunakan untuk menunjukkan dokumen *input* dan *output* baik untuk proses

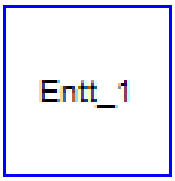
			manual, mekanik, atau komputer.
2		Keputusan	Simbol keputusan digunakan untuk menggambarkan suatu kondisi yang mengharuskan sistem untuk memilih tindakan yang akan dilakukan berdasarkan criteria tertentu.
3		Operasi *manual*	Simbol ini digunakan untuk menggambarkan proses yang terjadi secara *manual* yang tidak dapat dihilangkan dari sistem yang ada
4		*Database*	Simbol ini digunakan untuk menggambarkan media penyimpanan yang digunakan untuk menyimpan data pada sistem yang akan dibuat.
5		Proses	Simbol proses digunakan untuk menggambarkan proses yang terjadi dalam sistem yang akan dibuat
6		*Input* manual	Simbol Proses yang digunakan untuk menggambarkan proses yang terjadi dalam sistem yang akan dibuat.

2.6 *Data Flow Diagram (*DFD*)*

Menurut Kendall (2003: 241), *Data Flow Diagram* menggambarkan pandangan sejauh mungkin mengenai masukan, proses dan keluaran sistem, yang berhubungan dengan masukan, proses, dan keluaran dari model sistem yang dibahas. Serangkaian diagram aliran data berlapis juga bisa digunakan untuk merepresentasikan dan menganalisis prosedur-prosedur mendetail dalam sistem. Prosedur-prosedur tersebut yaitu konseptualisasi bagaimana data-data berpindah di dalam organisasi, proses-proses atau transformasi dimana data-data melalui, dan apa keluarannya. Jadi, melalui suatu teknik analisa data terstruktur yang disebut *Data Flow Diagram*, penganalisis sistem dapat merepresentasi proses-proses data di dalam organisasi. Menurut Kendall (2003: 265), dalam memetakan *Data Flow Diagram*, terdapat beberapa simbol yang digunakan antara lain:

1. *External entity*

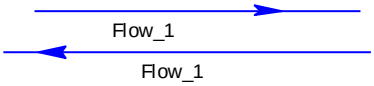
Suatu *external entity* atau entitas merupakan orang, kelompok, departemen, atau sistem lain di luar sistem yang dibuat dapat menerima atau memberikan informasi atau data ke dalam sistem yang dibuat.



Gambar 2.1 Simbol *External Entity*

2. *Data Flow*

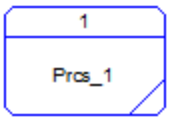
Data Flow atau aliran data disimbolkan dengan data tanda panah. Aliran data menunjukkan arus data atau aliran data yang menghubungkan dua proses atau *entitas* dengan proses.



Gambar 2.2 Simbol *Data Flow*

3. *Process*

Suatu proses dimana beberapa tindakan atau sekelompok tindakan dijalankan.



Gambar 2.3 Simbol *Process*

4. *Data Store*

Data store adalah simbol yang digunakan untuk melambangkan proses penyimpanan data.

Gambar 2.4 Simbol *Data Store*

2.7 *Entity Relationship Diagram* (ERD)

Entity relationship diagram (ERD) adalah gambaran pada sistem dimana di dalamnya terdapat hubungan antara *entity* beserta relasinya. *Entity* merupakan sesuatu yang ada dan terdefinisikan di dalam suatu organisasi, dapat abstrak dan nyata. Untuk setiap *entity* biasanya mempunyai *attribute* yang merupakan ciri *entity* tersebut. *Attribute* yaitu uraian dari entitas dimana mereka dihubungkan atau dapat dikatakan sebagai *identifier* atau *descriptors* dari entitas.

Entitas digolongkan menjadi *independent* atau *dependent entity*. *Independent entity* adalah apa yang tidak bersandar pada yang lain sebagai

identifikasi. Suatu *dependent entity* adalah apa yang bersandar pada yang lain sebagai identifikasi. Selain digolongkan menjadi *independent* atau *dependent entity*, terdapat jenis- jenis entitas khusus yaitu:

1. *Associative Entity*

Associative Entity (juga dikenal sebagai *intersection entity*) adalah entitas yang *digunakan* oleh rekanan dua entitas atau lebih untuk menyatukan suatu hubungan benyak - ke - banyak (*Many to Many*)

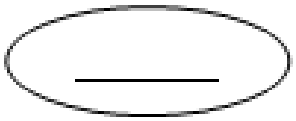
2. *Subtypes Entity*

Subtypes Entity digunakan di dalam hierarki generalisasi (*generalization hierarchies*) untuk menyajikan suatu subset kejadian dari entitas orangtua, yang disebut *supertype*, tetapi yang memiliki atribut atau hubungan yang berlaku hanya untuk *subset*.

Menurut Marlinda (2004: 28), *atribute* sebagai kolom di sebuah relasi mempunyai macam-macam jenis *atribute* yaitu :

a. *Key Atribute*

Atribute ini merupakan *atribute* yang unik dan tidak dimiliki oleh *atribute* lainnya, misalnya entity mahasiswa yang *atribute*-nya NIM.



Gambar 2.5 *Key Attribute*

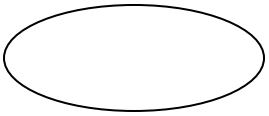
b. *Particial key Atribute*

Adalah *Attribute* yang tidak menjadi atau merupakan anggota dari *Key Primer*. Misalnya antara Cabang (toko) dan kode cabang.

Gambar 2.6 *Particial Key Attribute*

c. *Single Vallue Atribute*

Atribute yang hanya memiliki satu nilai harga, misalnya *entity* mahasiswa dengan *atribute*-nya Umur (Tanggal lahir).



Gambar 2.7 *Single Value Attribute*

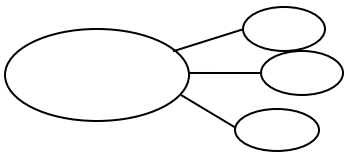
d. *Multi Vallue Atribute*

Atribute yang banyak memiliki nilai harga, misalnya *entity* mahasiswa dengan *atribute*-nya pendidikan (SD, SMP, SMA).

Gambar 2.8 *Multi Value Attribute*

e. *Composite Atribute*

Atribute yang memiliki dua harga, misalnya nama besar (nama kerja) dan nama kecil (nama asli)



Gambar 2.9 *Composite Attribute*

f. *Derived Attribute*

Attribute yang yang nilai-nilainya diperoleh dari pengolahan atau dapat diturunkan dari table *Attribute* atau table lain yang berhubungan.

Gambar 2.10 *Derived Attribute*

Model *Entity - Relationship* (ER) mula-mula diusulkan oleh Peter pada tahun 1976 sebagai cara untuk mempersatukan pandangan basis data jaringan dan relasional. Langkah sederhana dari model ER adalah model data konseptual yang memandang dunia nyata sebagai kesatuan (*entitias*) dan hubungan (*relationship*).

Komponen dasar model merupakan diagram *entity-relationship* yang digunakan untuk menyajikan objek data secara *visual*. *Entity Relationship Diagram* mengilustrasikan struktur logis dari basis data yang mempunyai metodologi sebagai berikut:

Tabel 2.2 Ilustrasi Pembuatan ERD

Proses	**Keterangan**
1. Menentukan Entitas	Menentukan peran, kejadian, lokasi, hal nyata, dan konsep dimana pengguna akan menyimpan data.
2. Menentukan Relasi	Tentukan hubungan antara pasangan entitas menggunakan matriks relasi.
3. Gambar ERD Sementara	Entitas digambarkan dengan kotak dan relasi dengan garis yang menghubungkan entitas.
4. Isi Kardinalitas	Tentukan jumlah kejadian dari satu entitas untuk sebuah kejadian pada

	entitas yang berhubungan.
5. Tentukan Kunci Utama	Tentukan atribut yang mengidentifikasi satu dan hanya satu kejadian pada masing-masing entitas.
6. Gambar ERD berdasar Kunci	Hilangkan relasi *Many-to-Many* dan masukkan *primary* dan kunci tamu pada masing-masing entitas.
7. Menentukan Atribut	Tuliskan *field-field* yang diperlukan oleh sistem.
8. Pemetaan Atribut	Pasangkan atribut dengan satu entitas yang sesuai pada masing-masing atribut.
9. Gambar ERD dengan Atribut	Aturlah ERD dari langkah 6 dengan menambahkan entitas atau relasi yang ditemukan pada langkah 8.
10. Periksa Hasil	Apakah ERD sudah menggambar sistem yang akan dibangun.

Entity Relationship Diagram ini diperlukan agar dapat menggambarkan hubungan antar *entity* dengan jelas, dapat menggambarkan batasan jumlah *entity* dan partisipasi antar *entity*, mudah dimengerti pemakai dan mudah disajikan oleh perandang *database*. Untuk itu, *entity relationship diagram* dibagi menjadi dua jenis model, yaitu:

1. *Conceptual Data model*

Conceptual Data model (CDM) adalah merupakan model yang *universal* dan dapat menggambarkan semua struktur *logic database* (DBMS), dan tidak

bergantung dari *software* atau pertimbangan struktur data *storage*. Sebuah CDM dapat diubah langsung menjadi PDM.

2. *Physical Data Model*

Physical Data Model (PDM) adalah merupakan model ERD yang telah mengacu pada pemilihan *software* DBMS yang spesifik. Hal ini sering kali berbeda dikarenakan oleh struktur *database* yang bervariasi, mulai dari model *schema*, tipe data penyimpanan dan sebagainya.

2.8 MySQL

MySQL adalah sebuah perangkat lunak *database* (basis data) sistem terbuka yang sangat terkenal di kalangan pengembang sistem *database* dunia yang digunakan untuk berbagai aplikasi terutama untuk aplikasi berbasis web. MySQL mempunyai fungsi sebagai SQL (*Structured Query Language*) telah diperluas. MySQL umumnya digunakan bersama dengan PHP untuk membuat aplikasi yang dinamis dan powerful.

2.9 PHP (*Hypertext Preprocessor*)

Menurut Saputra (2011, p.1) PHP atau yang memiliki kepanjangan PHP *Hypertext Preprocessor* merupakan suatu bahasa pemrograman yang difungsikan untuk membangun suatu website dinamis. PHP menyatu dengan kode HTML, maksudnya adalah beda kondisi. HTML digunakan sebagai pembangun atau pondasi dari kerangka layout web, sedangkan PHP difungsikan sebagai prosesnya sehingga dengan adanya PHP tersebut, web akan sangat mudah di-*maintenance*. PHP berjalan pada sisi *server* sehingga PHP disebut juga sebagai bahasa *server Side Scripting*. Artinya bahwa dalam setiap menjalankan PHP, wajib adanya web server.

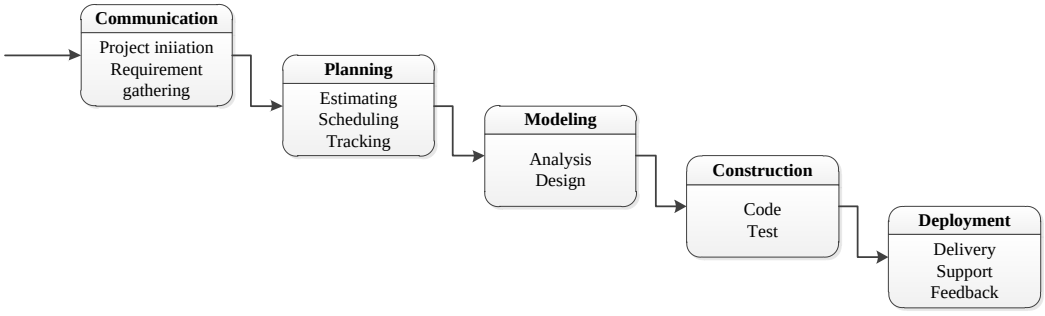
2.10 HTML (*HyperText Markup Language*)

HTML (*HyperText Markup Language*) adalah sebuah bahasa standar yang digunakan oleh *browser* internet untuk membuat halaman dan dokumen pada sebuah *web* yang kemudian dapat diakses dan dibaca layaknya sebuah artikel. HTML juga dapat digunakan sebagai penghubung antara file-file dalam situs atau dalam komputer dengan menggunakan *localhost*.

2.11 Siklus Hidup Pengembangan Sistem (SDLC)

Menurut pressman (2001), Model *System Development Life Cycle (SDLC)* ini biasa disebut juga dengan model *waterfall* atau disebut juga *classic life cycle.* Adapun pengertian dari SDLC ini adalah suatu pendekatan yang sistematis dan berurutan. Tahapan-tahapannya adalah *Requirements* (analisis sistem), *Analysis* (analisis kebutuhan sistem), *Design* (perancangan), *Coding* (implementasi), *Testing* (pengujian) dan *Maintenance* (perawatan).

Model eksplisit pertama dari proses pengembangan perangkat lunak, berasal dari proses-proses rekayasa yang lain. Model ini memungkinkan proses pengembangan lebih terlihat. Hal ini dikarenakan bentuknya yang bertingkat ke bawah dari satu fase ke fase lainnya, model ini dikenal dengan model *waterfall*, seperti pada gambar 2.11



Gambar 2.11 Bagan *System Development Life Cycle* (SDLC) Model *Waterfall*

Gambar 2.11 menunjukkan tahapan umum dari model proses *waterfall*. Model ini disebut dengan *waterfall* karena tahap demi tahap yang dilalui harus menunggu selesainya tahap sebelumnya dan berjalan berurutan. Akan tetapi, Pressman (2015) memecah model ini meskipun secara garis besar sama dengan tahapan-tahapan model *waterfall* pada umumnya.

Model ini merupakan model yang paling banyak dipakai dalam *Software Engineering*. Model ini melakukan pendekatan secara sistematis dan urut mulai dari *level* kebutuhan sistem lalu menuju ke tahap *Communication, Planning, Modeling, Construction,* dan *Deployment*.

Berikut ini adalah penjelasan dari tahap-tahap yang dilakukan di dalam Model *Waterfall* menurut Pressman (2015):

a. *Communication*

Langkah pertama diawali dengan komunikasi kepada konsumen/pengguna. Langkah awal ini merupakan langkah penting karena menyangkut pengumpulan informasi tentang kebutuhan konsumen/pengguna.

b. *Planning*

Setelah proses *communication* ini, kemudian menetapkan rencana untuk pengerjaan *software* yang meliputi tugas-tugas teknis yang akan dilakukan, risiko yang mungkin terjadi, sumber yang dibutuhkan, hasil yang akan dibuat, dan jadwal pengerjaan.

c. *Modeling*

Pada proses modeling ini menerjemahkan syarat kebutuhan ke sebuah perancangan perangkat lunak yang dapat diperkirakan sebelum dibuat *coding*.

Proses ini berfokus pada rancangan struktur data, arsitektur *software*, representasi *interface*, dan detail (algoritma) prosedural.

d. *Construction*

Construction merupakan proses membuat kode (*code generation*). *Coding* atau pengkodean merupakan penerjemahan desain dalam bahasa yang bisa dikenali oleh komputer. Programmer akan menerjemahkan transaksi yang diminta oleh user. Tahapan inilah yang merupakan tahapan secara nyata dalam mengerjakan suatu *software*, artinya penggunaan komputer akan dimaksimalkan dalam tahapan ini. Setelah pengkodean selesai maka akan dilakukan testing terhadap sistem yang telah dibuat. Tujuan testing adalah menemukan kesalahan-kesalahan terhadap sistem tersebut untuk kemudian bisa diperbaiki.

e. *Deployment*

Tahapan ini bisa dikatakan final dalam pembuatan sebuah *software* atau sistem. Setelah melakukan analisis, desain dan pengkodean maka sistem yang sudah jadi akan digunakan *user*. Kemudian *software* yang telah dibuat harus dilakukan pemeliharaan secara berkala.

2.12 *Testing*

Menurut Romeo (2003), testing adalah proses pemantapan kepercayaan akan kinerja program atau sistem sebagaimana yang diharapkan. Testing *Software* adalah proses pengoperasikan *software* dalam suatu kondisi yang dikendalikan untuk verifikasi, mendeteksi *error* dan validasi. Verifikasi adalah pengecekkan atau pengetesan entitas-entitas, termasuk *software*, untuk pemenuhan dan konsistensi dengan melakukan evaluasi hasil terhadap kebutuhan yang telah ditetapkan. Validasi adalah melihat kebenaran sistem apakah proses yang telah

dituliskan sudah sesuai dengan apa yang dibutuhkan oleh pengguna. Deteksi *error* adalah testing yang berorentasi untuk membuat kesalahan secara intensif, untuk menentukan apakah suatu hal tersebut tidak terjadi. *Test case* merupakan suatu tes yang dilakukan berdasarkan pada suatu inisialisasi, masukan, kondisi ataupun hasil yang telah ditentukan sebelumnya. Adapun kegunaan dari *test case* ini, adalah sebagai berikut:

1. Untuk melakukan testing kesesuaian suatu komponen terhadap desain *White Box Testing*.
2. Untuk melakukan testing kesesuaian suatu komponen terhadap spesifikasi *Black Box Testing*.

2.12.1 *White Box Testing*

Menurut Romeo (2003), *white box testing* adalah suatu metode desain *test case* yang menggunakan struktur kendali dari desain prosedural. Seringkali *white box testing* diasosiasikan dengan pengukuran cakupan tes, yang mengukur persentase jalur-jalir dari tipe yang dipilih untuk dieksekusi oleh *test cases*. *White box testing* dapat menjamin semua struktur *internal* data dapat dites untuk memastikan validitasnya.

Cakupan pernyataan, cabang dan jalur adalah suatu teknik *white box testing* yang menggunakan alur logika dari program untuk membuat *test cases* alur logika adalah cara dimana suatu bagian dari program tertentu dieksekusi saat menjalankan program. Alur logika suatu program dapat direpresentasikan dengan *flow graph.*

2.12.2 *Black Box Testing*

Menurut Romeo (2003), *Black box testing* dilakukan tanpa adanya suatu pengetahuan tentang detail struktur internal dari sistem atau komponen yang dites, juga disebut sebagai *functional testing*. *Black box testing* bergfokus pada kebutuhan fungsional pada *software*, berdasarkan pada spesifikasi kebutuhan dari *software*.

Dengan adanya *black box testing*, perekayasa *software* dapat menggunakan kebutuhan fungsional pada suatu program. *Black box testing* dilakukan untuk melakukan pengecekan apakah sebuah *software* telah bebas dari *error* dan fungsi-fungsi yang diperlukan telah berjalan sesuai dengan yang diharapkan.