

## **BAB II**

### **LANDASAN TEORI**

#### **2.1. Perancangan**

O'Brien dan Marakas (2009:639) menjelaskan bahwa perancangan sistem adalah sebuah kegiatan merancang dan menentukan cara mengolah sistem informasi dari hasil analisa sistem sehingga dapat memenuhi kebutuhan dari pengguna termasuk diantaranya perancangan user interface, data dan aktifitas proses.

Menurut Bentley dan Whitten (2009:160) menjelaskan bahwa perancangan sistem adalah teknik pemecahan masalah dengan melengkapi komponen – komponen kecil menjadi kesatuan komponen sistem kembali ke sistem yang lengkap. Teknik ini diharapkan dapat menghasilkan sistem yang lebih baik.

Menurut Satzinger, Jackson dan Burd (2012:5) menjelaskan perancangan adalah sekumpulan aktifitas yang menggambarkan secara rinci bagaimana sistem akan berjalan. Hal itu bertujuan untuk menghasilkan produk perangkat lunak yang sesuai dengan kebutuhan user.

Menurut Bentley dan Whitten (2009:160) menjelaskan bahwa perancangan sistem adalah teknik pemecahan masalah dengan melengkapi komponen – komponen kecil menjadi kesatuan komponen sistem kembali ke sistem yang lengkap. Teknik ini diharapkan dapat menghasilkan sistem yang lebih baik.

Berdasarkan pengertian perancangan menurut para ahli diatas, dapat disimpulkan bahwa perancangan adalah suatu proses untuk membuat dan mendesain suatu sistem yang baru.

## **2.2. Implementasi**

Menurut Kusumanegara (2010:97) Implementasi adalah sebuah tahapan yang dilakukan setelah aturan hukum ditetapkan melalui proses politik. Kalimat tersebut seolah-olah menunjukkan bahwa implementasi lebih bermakna non politik, yaitu administratif. Pandangan Lester dan Stewart juga Kusumanegara di atas, bahwa dalam proses implementasi perlu beberapa tahapan atau alur proses yang didalamnya tercakup keterlibatan berbagai macam aktor, organisasi, prosedur agar kebijakan yang telah ditetapkan mempunyai akibat, yaitu tercapainya tujuan kebijakan.

## **2.3. Sistem Pakar**

Menurut B. Hermawan Hayadi (2018:1) Sistem pakar biasa disebut juga dengan *Knowledge Based System* yaitu suatu aplikasi komputer yang ditunjukan untuk membantu pengambilan keputusan atau pemecahan persoalan dalam bidang yang spesifik. Sistem ini berkerja dengan menggunakan pengetahuan dan metode analisis yang telah didefinisikan terlebih dahulu oleh pakar yang sesuai dengan bidang keahliannya. Sistem ini disebut sistem pakar karena fungsi dan perannya sama seperti seorang ahli yang harus memiliki pengetahuan, pengalaman dalam memecahkan suatu persoalan. Sistem biasanya berfungsi sebagai kunci penting yang akan membantu suatu sistem pendukung keputusan atau sistem pendukung eksekutif.

## **2.4. Diagnosa**

Menurut Lynda Jual Carpenito (2018:67) diagnosa adalah penilaian klinik tentang respon individu, keluarga atau komunitas terhadap masalah kesehatan/ proses kehidupan yang aktual atau potensial. Diagnosa memberikan dasar untuk pemilihan intervensi keperawatan untuk mencapai hasil yang merupakan tanggung jawab (Disetujui oleh anggota *North American Nursing Diagnosis Association (NANDA)*, Konferensi IX, Maret 1990).

## 2.5. Penyakit

Menurut Sudoyo Aru. W (2009:63) penyakit adalah kondisi dimana tidak berfungsinya sistem ini, yaitu suatu penyimpangan dari keadaan tubuh yang normal atau ketidak harmonisan jiwa. Menurut konsep ini, penyakit mempunyai fungsi negatif dan terbatas pada dimensi fisik. Pengertian penyakit yang seperti ini memandang penyakit sebagai suatu penyimpangan dari manusia ideal dan sebagai suatu ancaman bagi kebahagiaan manusia dan kenikmatan hidup yang utuh.

## 2.6. Tulang Belakang

Menurut Evelyn C. Pearce (2009:66) *Kolumna vertebralis* atau rangkaian tulang belakang adalah struktur lentur sejumlah tulang yang disebut *vertebrata* atau ruas tulang belakang. Di antara tiap dua ruas tulang pada tulang belakang terdapat bantalan tulang rawan. Panjang rangkaian tulang belakang pada orang dewasa dapat mencapai 57 sampai 67 sentimeter. Seluruhnya terdapat 33 ruas tulang, 24 buah diantaranya adalah tulang – tulang terpisah dan 9 ruas sisanya bergabung membentuk 2 tulang.

*Vertebra* dikelompokkan dan dinamai sesuai dengan daerah yang ditempatinya :

1. Tujuh *vertebra servikal* atau ruas tulang leher membentuk daerah tengkuk/
2. Dua belas *vertebra torakalis* atau ruas tulang punggung membentuk bagian belakang *toraks* atau dada.
3. Lima *vertebra lumbalis* atau ruas tulang pinggang membentuk daerah *lumbal* atau pinggang
4. Empat *vertebra koksigeus* atau ruas tulang tungging membentuk tulang *koksigeus* atau tulang tungging.

## 2.7. Website atau Situs

Menurut Rahmat Hidayat (2010:1) *Website* atau situs dapat diartikan sebagai kumpulan halaman – halaman yang digunakan untuk menampilkan informasi teks, gambar diam atau gerak, animasi, suara dan atau gabungan dari

semuanya, baik yang bersifat statis maupun dinamis yang membentuk satu rangkaian bangunan yang saling terkait, yang masing – masing dihubungkan dengan jaringan – jaringan halaman. Hubungan antara satu halaman web dengan halaman web yang lainnya disebut *Hyperlink*, sedangkan teks yang dijadikan media pengubung disebut *Hypertext*.

Ada beberapa hal yang dipersiapkan untuk membangun website, maka harus tersedia unsur – unsur pendukungnya sebagai berikut :

1. Nama Domain (Domain name/*URL Uniform Resorce Locator*)
2. Rumah Website (*Website Hosting*)
3. *Content Management System (CMS)*

Perkembangan dunia *website* pada saat ini lebih menekankan pada pengelolaan konten sebuah *website*. Pengguna yang tidak bisa bahasa pemrograman *website* pada saat ini bisa membuat *website* dengan memanfaatkan *CMS* tersebut.

#### **2.4.1 Jenis – Jenis Web**

Menurut Rahmat Hidayat (2010:3) Seiringan dengan perkembangan teknologi informasi yang begitu cepat, *website* juga mengalami perkembangan yang sangat berarti. Dalam pengelompokan jenis *web*, lebih diarahkan berdasarkan kepada fungsi, sifat atau *style* dan bahasa pemrograman yang digunakan.

1. *Website* Dinamis, merupakan sebuah *website* yang menyediakan konten atau isi yang selalu berubah – ubah setiap saat. Bahasa pemrograman yang digunakan antara lain PHP, ASP, .NET dan memanfaatkan *database* MySQL atau MS SQL.
2. *Website* statis, merupakan *website* yang kontennya sangat jarang diubah, Bahasa pemrograman yang digunakan adalah HTML dan belum memanfaatkan *database*.

Berdasarkan pada fungsinya, *website* terbagi atas :

1. Personal *website*, *website* yang berisi informasi pribadi seseorang.

2. *Commercial Website*, *website* yang dimiliki oleh sebuah perusahaan yang bersifat bisnis.
3. *Government Website*, *website* yang dimiliki oleh instansi pemerintahan, pendidikan yang bertujuan memberikan pelayanan kepada pengguna.
4. *Non-Profit Organization Website*, *website* yang dimiliki oleh organisasi yang bersifat *non-profit* atau tidak bersifat bisnis.

#### 2.4.2 Pengertian Domain

Menurut Rahmat Hidayat (2010:9) Domain adalah alamat unik di dunia internet yang digunakan untuk mengidentifikasi sebuah *website*, atau dengan kata lain domain adalah alamat yang digunakan untuk mencari dan menemukan sebuah *website* pada dunia *internet*. Nama domain diperjual belikan secara bebas di internet dengan status sewa tahunan. Nama domain sendiri mempunyai identifikasi ekstensi/akhiran sesuai dengan kepentingan dan lokasi keberadaan *website* tersebut. Apabila dibahas lebih lanjut mengenai pengertian domain, kita perlu sedikit mengetahui tentang bagaimana sebuah host di lingkungan internet diakses. Internet terdiri dari jutaan computer sebagai *host* yang tersebar di seluruh dunia yang semuanya saling bergubungan melalui suatu bentuk jaringan dengan hirarki tertentu. *Host – host* tersebut saling berkomunikasi melalui suatu protocol standar yang disebut TCP/IP (*Transmission Control Protocol/Internet Protocol*). Agar setiap computer yang membentuk jaringan internet dapat berkomunikasi satu sama lain, masing – masing haruslah memiliki alamat tertentu. Alamat ini haruslah unik, jadi tidak boleh ada dua *host* yang memiliki alamat yang sama.

Sistem pengalamatan yang digunakan berupa kombinasi 4 deret bilangan antara 0 s/d 255 yang masing – masing dipisahkan oleh tanda titik (.), mulai dari 0.0.0.1 hingga 255.255.255.255.

Deretan angka – angka ini dikenal sebagai alamat IP (*IP address*). Setiap host yang tersambung dalam jaringan internet harus memiliki alamat IP sebagai pengenalan agar dapat berkomunikasi dengan host lain dalam jaringan. Pengalamatan berbasis IP ini memungkinkan internet mengalami lebih dari 4 milyar host. Pada kenyataannya, tidak semua kombinasi alamat IP bisa dipergunakan. Ada beberapa kombinasi khusus yang dicadangkan untuk keperluan tertentu sehingga tidak boleh digunakan untuk keperluan pengalamatan. Contohnya IP 127.0.0.1 yang diperlukan untuk menunjuk (*lookup*) ke host lokal. Walaupun secara teknis sistem pengalamatan berbasis IP cukup handal, tetapi masih memiliki kelemahan. Otak manusia umumnya tidak mudah menghafal dan mengingat kombinasi angka dalam jumlah besar. Solusinya adalah mengasosiasikan nomor IP dalam kombinasi huruf yang membentuk sebuah nama yang mudah diingat. Nama host sebagai pengenalan jaringan internet inilah yang disebut sebagai domain, sedangkan sistem pengalamatan berbasis domain dikenal sebagai *Domain Name Service (DNS)*.

## **2.8. Rekayasa Perangkat Lunak**

Menurut Soetam Rizky Wicaksono(2017:24) Secara teoritis, banyak sekali definisi mengenai RPL, baik yang berasal dari buku teks maupun lembaga mandiri seperti ACM dan IEEE atau juga dari sumber internet. Berikut ini adalah beberapa definisi resmi dari RPL atau *software engineering* yang diambil dari beberapa sumber yakni

1. Dari Sommerville :

*“Software engineering is an engineering discipline which is concerned with all aspects of software production from the early stage of system after it has gone into use.”*

Jadi RPL merupakan sebuah disiplin ilmu yang berhubungan dengan seluruh aspek produk perangkat lunak baik dari tahap awal hingga ke pemeliharaan dari perangkat lunak pasca produksi. Dari

definisi ini tersirat jelas bahwa RPL tidak hanya berkutat pada masalah perencanaan dan perancangan yang hamper selalu menjadi kesalahpahaman dalam proses pengajaran mata kuliah RPL. Sehingga RPL sendiri adalah sebuah proses yang terintegrasi dan menyeluruh dari segala aspek, mulai dari sebelum perangkat lunak itu dibuat hingga selesai dan bahkan hingga tahap penggunaan.

2. Dari IEEE :

*“Software engineering is defined as the application of systematic, discipline, quantifiable, approach to development, operation and maintenance software.”*

Definisi yang kedua menyebutkan bahwa RPL selain sistematis juga merupakan pendekatan yang seharusnya mampu untuk di kuantifikasikan atau diukur keberadaannya dengan angka – angka atau ukuran tertentu dalam sebuah proses pengembangan perangkat lunak.

3. Dari Conger :

*“Software engineering is the systematic development, operation, maintenance, and retirement of software.”*

Definisi ketiga juga menyatakan bahwa RPL adalah sebuah proses yang sistematis, bahkan hingga ke proses saat perangkat lunak tidak lagi digunakan.

4. Dari Gezli et al :

*“Software engineering is the field of computer science that deals with the building of software system that are so large or so complex that they are built by a team or teams of engineers.”*

Dari definisi yang keempat terlihat bahwa penekanan dari RPL hanya terjadi jika sebuah perangkat lunak dianggap telah memiliki skala besar dan dianggap kompleks, sehingga membutuhkan sebuah tim untuk mengerjakannya. Tentu saja definisi ini sedikit bertentangan dengan definisi – definisi sebelumnya, karena di

definisi sebelumnya telah disebutkan bahwa RPL tidak memiliki hubungan dengan asas skalabilitas (besar kecilnya sebuah perangkat lunak), tetapi lebih ke arah sebuah siklus hidup dari mulai perancangan hingga pemeliharaan.

5. Dari Bjorner :

*“Software engineering is the establishment and use of sound methods for the efficient construction of efficient, correct, timely and pleasing software that solves the problems such as user identify them.”*

Dari definisi yang terakhir ini, disebutkan bahwa RPL lebih bertujuan ke arah sebuah efisiensi pengerjaan perangkat lunak dan mampu memuaskan pengguna dengan asumsi telah mampu memecahkan masalah yang dihadapi oleh pengguna.

### **2.5.1. Lingkup Rekayasa Perangkat Lunak**

Menurut Soetam Rizky Wicaksono(2017:48) Seperti telah dijelaskan sebelumnya, bahwa RPL bukanlah sekedar sebuah proses perancangan atau analisa dengan mempelajari secara detail dan mendalam mengenai teori – teori analisa dan perancangan belaka. Tetapi juga melebar hingga teori pemeliharaan perangkat lunak pasca produksi.

Dari beberapa buku teks, telah dijelaskan dengan sangat gamblang bahwa RPL meliputi beberapa pokok bahasan penting antara lain :

1. Domain Engineering

Mampu memahami permasalahan yang muncul dan akan dijadikan sebagai proyek perangkat lunak

2. Requirement Engineering

Mampu memahami kebutuhan pengguna sekaligus melakukan pemecahan masalah

3. Software Design



Mampu memahami serta mengimplementasikan perancangan perangkat lunak termasuk didalamnya aspek HCI (*Human Computer Interaction*)

#### 4. *Development*

Dalam proses pengembangan sebuah perangkat lunak nantinya akan melibatkan pembelajaran mengenai algoritma, bahasa pemrograman yang digunakan serta teknik yang berkaitan didalamnya seperti basis data dan sistem informasi.

#### 5. *Operations*

Operasional perangkat lunak dapat dipisahkan menjadi dua bagian penting yakni pada saat proses testing yang seharusnya dilakukan bersama – sama antara pengembang dan pengguna perangkat lunak, dan proses implementasi yang didalamnya bisa terdapat langkah – langkah awal seperti pelatihan dan perbaikan pasca produksi.

#### 6. *Maintenance*

Dalam ruang lingkup yang terakhir ini selain melakukan pemeliharaan terhadap aspek perangkat lunak seperti basis data, instalasi juga didalamnya terdapat proses dokumentasi dari pengembang perangkat lunak. Proses dokumentasi nantinya tidak hanya ditujukan untuk pengguna, tetapi juga untuk pengembang perangkat lunak itu sendiri jika nantinya terjadi proses revisi atau penggantian lebih lanjut.

Sedangkan jika dilihat dari proses sebuah RPL sendiri melibatkan beberapa unsur antara lain :

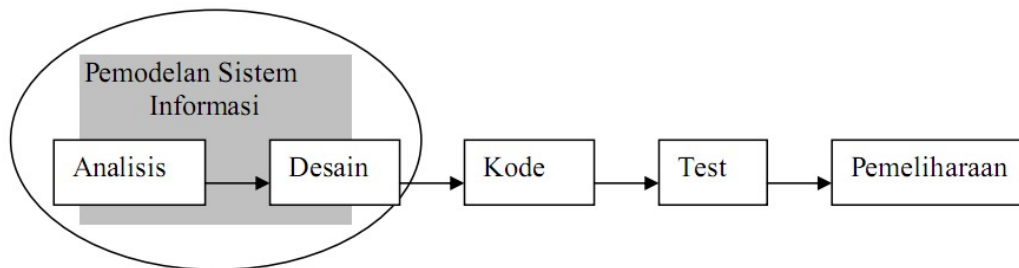
1. Software Engineers atau pengembang perangkat lunak
2. Software atau perangk lunak
3. User atau pengguna

## 2.9. Model Proses Dalam Rekayasa Perangkat Lunak

Menurut Pressman (2015:42), model *waterfall* adalah model klasik yang bersifat sistematis, berurutan dalam membangun *software*. Nama model ini

sebenarnya adalah “*Linear Sequential Model*”. Model ini sering disebut juga dengan “*classic life cycle*” atau metode *waterfall*. Model ini termasuk ke dalam model *generic* pada rekayasa perangkat lunak dan pertama kali diperkenalkan oleh Winstin Royce sekita tahun 1970 sehingga sering dianggap kuno, tetapi merupakan model yang paling banyak dipakai dalam *Software Engineering* (SE). Model ini melakukan pendekatan secara sistematis dan berurutan. Disebut dengan *waterfall* karena tahap demi tahap yang dilalui harus menunggu selesainya tahap sebelumnya dan berjalan berurutan.

Dengan demikian, metode *waterfall* dianggap pendekatan yang lebih cocok digunakan untuk proyek pembuatan sistem baru dan juga pengembangan *software* dengan tingkat resiko yang kecil serta waktu pengembangan yang cukup lama. Tetapi salah satu kelemahan paling mendasar adalah menyamakan pengembangan *hardware* dan *software* dengan meniadakan perubahan saat pengembangan. Padahal, *error* diketahui saat *software* dihalankan, dan perubahan – perubahan akan sering terjadi.



**Gambar 2.4** *Waterfall*

### **1. Rekayasa dan pemodelan sistem/informasi.**

Karena perangkat lunak selalu merupakan bagian dari sebuah sistem yang lebih besar, kerja dimulai dengan membangun syarat dari semua elemen sistem dan mengalokasikan beberapa subset dari kebutuhan ke perangkat lunak tersebut.

### **2. Analisis kebutuhan perangkat lunak.**

Proses pengumpulan kebutuhan diintensifkan dan difokuskan, khususnya pada perangkat lunak. Untuk memahami sifat program yang dibangun, perekrayasa perangkat lunak (*analisis*) harus memahami

domain informasi, tingkah laku, unjuk kerja, dan antar muka (*interface*) yang diperlukan. Kebutuhan baik untuk sistem maupun perangkat lunak didokumentasikan dan dilihat lagi dengan pelanggan.

### **3. Desain.**

Desain perangkat lunak sebenarnya adalah proses multi langkah yang berfokus pada empat atribut sebuah program yang berbeda; struktur data, arsitektur perangkat lunak, representasi *interface*, dan detail (algoritma) prosedural. Proses desain menerjemahkan syarat/kebutuhan ke dalam sebuah representasi perangkat lunak yang dapat diperkirakan demi kualitas sebelum dimulai pemunculan kode. Sebagaimana persyaratan, desain didokumentasikan dan menjadi bagian dari konfigurasi perangkat lunak.

### **4. Kode.**

Desain harus diterjemahkan ke dalam bentuk mesin yang bisa dibaca. Langkah pembuatan kode melakukan tugas ini. Jika desain dilakukan dengan cara yang lengkap, pembuatan kode dapat diselesaikan secara mekanis.

### **5. Test.**

Sekali kode dibuat, pengujian program dimulai. Proses pengujian berfokus pada logika internal perangkat lunak, memastikan bahwa semua pernyataan sudah diuji, dan pada eksternal fungsional – yaitu mengarahkan pengujian untuk menemukan kesalahan-kesalahan dan memastikan bahwa input yang dibatasi akan memberikan hasil aktual yang sesuai dengan hasil yang dibutuhkan.

### **6. Pemeliharaan.**

Perangkat lunak akan mengalami perubahan setelah disampaikan kepada pelanggan. Perubahan akan terjadi karena kesalahan-kesalahan ditentukan, karena perangkat lunak harus disesuaikan untuk mengakomodasi perubahan-perubahan di dalam lingkungan eksternalnya, atau karena pelanggan membutuhkan perkembangan fungsional atau unjuk kerja. Pemeliharaan perangkat

lunak mengaplikasikan lagi setiap fase program sebelumnya dan tidak membuat yang baru lagi.

## 2.10. Unified Modelling Language (UML)


Menurut Gandharba Swain (2010:24) *UML* adalah bahasa kosa kata dan aturannya memfokuskan representasi konseptual dan fisik dari suatu sistem. Sistem intensif peragkat lunak memerlukan bahasa yang membahas berbagai pandangan arsitektur sistem. Kosa kata dan aturan bahasa seperti UML memberi model yang baik, tetapi tidak memberi tahu model apa yang harus anda buat dan kapan harus dibuat. Proses yang terdefinisi dengan baik akan memandu dalam memutuskan artefak apa yang akan diproduksi, kegiatan apa dan apa yang digunakan pekerjaan untuk menciptakannya dan mengelolanya, dan bagaimana menggunakan artefak untuk menggunakan artefak untuk mengukur dan mengendalikan proyek secara keseluruhan.

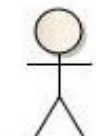

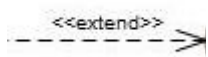
### 2.8.1. Use Case Diagram


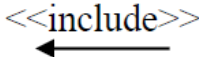
Menurut Gandharba Swain (2010:59) *Use case diagram* adalah sekumpulan urutan aksi yang melakukan hasil yang dapat diamati oleh aktor. Secara grafis *use case* diwakili oleh elips. Setiap *use case* harus memiliki nama yang membedakannya dari *use case* lainnya. Bisa berupa nama atau nama jalur sederhana. Dalam nama path, nama *use case* harus diawali dengan nama paket itu sendiri.

Simbol-simbol yang digunakan pada *use case diagram* ditunjukkan pada tabel berikut.

**Tabel 2.1** Simbol-simbol pada *Use Case diagram*

| Nama            | Simbol  | Deskripsi   |
|-----------------|---|---|
| <i>Use Case</i> |  | Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal di frase nama <i>Use Case</i> . |

|          |   |   |
|----------|---|---|
| Aktor    |    | Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri. Aktor hanya memberikan informasi ke sistem, aktor hanya menerima informasi dari sistem, aktor memberikan dan menerima informasi ke sistem dan dari sistem.  |
| Asosiasi |    | Komunikasi antara aktor dan <i>use case</i> yang berpartisipasi pada <i>use case</i> atau <i>use case</i> memiliki interaksi dengan aktor. Asosiasi merupakan hubungan statis antar elemen yang menggambarkan elemen yang memiliki atribut berupa elemen lain, atau elemen yang harus mengetahui eksistensi elemen lain.  |
| Ekstensi |  | Relasi <i>use case</i> tambahan ke sebuah <i>use case</i> dimana <i>use case</i> yang ditambahkan dapat berdiri sendiri walaupun tanpa <i>use case</i> tambahan itu, mirip dengan prinsip <i>inheritance</i> pada pemrograman berorientasi objek. Biasanya <i>use case</i> tambahan memiliki nama depan yang sama dengan <i>use case</i> yang ditambahkan. Misalnya arah panah mengarah pada <i>use case</i> yang ditambahkan, biasanya <i>use case</i> yang menjadi <i>extend</i> -nya merupakan jenis yang sama dengan <i>use case</i> yang menjadi induknya. |

|              |   |  |
|--------------|---|--|
| Generalisasi |  | Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah <i>use case</i> dimana fungsi yang satu adalah fungsi yang lebih umum dari yang lainnya, misalnya : arah panah mengarah pada <i>use case</i> yang menjadi generalisasinya (umum). Generalisasi merupakan hubungan hirarkis antara elemen. Elemen dapat mewarisi semua atribut dan metode elemen asalnya dan menambah fungsionalitas baru. |
| Include      |  | Relasi <i>use case</i> tambahan ke sebuah <i>use case</i> yang ditambahkan memerlukan <i>use case</i> ini untuk menjalankan fungsinya atau sebagai syarat.   |

### 2.8.2. Sequence Diagram

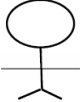

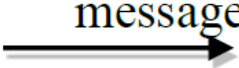

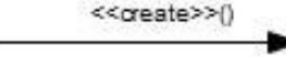
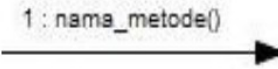
Menurut John Sarzinger, 2010, dalam buku *System Analysis and Design in a Changing World*, “*System Squence Diagram (SSD)* adalah diagram yang digunakan untuk mendefinisikan *input* dan *output* serta urutan interaksi antara pengguna dan sistem untuk sebuah *use case*”.

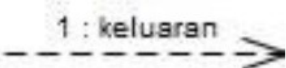
Menurut Gandharba Swain (2010:59) *Sequence diagram* adalah diagram yang memuat interaksi yang menekankan urutan waktu pesan. *Sequence diagram* menunjukkan serangkaian objek dan pesan yang dikirim dan diterima oleh objek – objek tersebut. Objek biasanya bernama atau anonim untuk contoh kelas, tetapi juga dapat mewakili contoh dari hal – hal lain, seperti kolaborasi, komponen , dan node, *sequence diagram* digunakan untuk menggambarkan tampilan dinamis suatu sistem.

Simbol – simbol yang digunakan pada *sequence diagram* ditunjuka Tabel

2.2

**Tabel 2.2** Simbol-simbol pada *sequence diagram*

| Nama                     | Simbol  | Deskripsi   |
|--------------------------|---|---|
| Aktor                    |    | Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri.                 |
| Lifeline                 |    | Menyatakan kehidupan suatu objek, untuk menggambarkan kelas dan objek.  |
| Objek                    |    | Menyatakan objek yang berinteraksi (pesan).   |
| Waktu Aktif              |   | Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif.   |
| Pesan tipe <i>create</i> |  | Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat.   |
| Pesan tipe <i>call</i>   |  | Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri, sesuai dengan kelas objek yang berinteraksi.                       |
| Pesan tipe <i>return</i> |   | Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang |


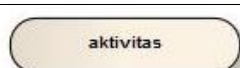



|  |   |                     |
|--|---|---------------------|
|  |  | menerima kembalian. |
|--|---|---------------------|

### 2.8.3. Activity Diagram

Menurut Gandharba Swain (2010:24) *Activity diagram* menunjukkan aliran dari aktivitas ke aktivitas dalam suatu sistem. Suatu aktivitas menunjukkan serangkaian aktivitas, aliran berurutan atau bercabang dari aktivitas ke aktivitas, dan objek yang bertindak dan ditindak lanjuti. *Activity diagram* sangat penting dalam memodelkan fungsi suatu sistem. *Activity diagram* menekankan aliran kontrol diantara objek.

Simbol-simbol yang digunakan pada *activity diagram* ditunjukkan pada tabel dibawah ini.

**Tabel 2.3** Simbol-simbol pada *activity diagram*

| Nama                            | Simbol  | Deskripsi   |
|---------------------------------|---|---|
| Status Awal                     |  | Status awal aktivitas sistem, sebuah aktivitas memiliki sebuah status awal                |
| Aktivitas                       |  | Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.            |
| percabangan/<br><i>decision</i> |  | Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.                   |
| Penggabungan<br><i>/ join</i>   |  | Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.          |
| Status akhir                    |  | Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir |



|          |                          |  |
|----------|--------------------------|--|
| Swimlane | <div>nama swimlane</div> | Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi. |
|----------|--------------------------|--|

### 2.11. Entity Relationship Diagram

Menurut Sutanta (2011:91) “*Entity Relationship Diagram* (ERD) merupakan suatu model data yang dikembangkan berdasarkan objek.” *Entity Relationship Diagram* (ERD) digunakan untuk menjelaskan hubungan antar data dalam basis data kepada pengguna secara logis. *Entity Relationship Diagram* (ERD) didasarkan pada suatu persepsi bahwa *real world* terdiri atas obyek – obyek dasar tersebut. Penggunaan *Entity Relationship Diagram* (ERD) relative mudah dipahami, bahkan oleh para pengguna yang awam. Bagi perancang atau analis sistem, *Entity Relationship Diagram* (ERD) berguna untuk memodelkan sistem yang nantinya, basis data akan dikembangkan. Model ini juga membantu perancangan atau analis sistem pada saat melakukan analis dan perancangan basis data karena model ini dapat menunjukkan macam data yang dibutuhkan dan kerelasiaan antar data didalamnya.

#### 2.9.1. Komponen Entity Relationship Diagram (ERD)

Komponen *Entity Relationship Diagram* (ERD) menurut Sutanta (2011:91) adalah sebagai berikut:

1. Entitas merupakan suatu obyek yang dapat dibedakan dari lainnya yang dapat diwujudkan dalam basis data. Obyek dasar dapat berupa orang, benda atau hal yang keterangannya perlu disimpan di dalam basis data. Untuk menggambarkan sebuah entitas digunakan aturan sebagai berikut:
  - 1) Entitas dinyatakan dengan simbol persegi panjang.
  - 2) Nama entitas dituliskan didalam simbol persegi panjang.
  - 3) Nama entitas berupa kata benda, tunggal.

- 4) Nama entitas sedapat mungkin menggunakan nama yang mudah dipahami dan dapat menyatakan maknanya dengan jelas.
2. Atribut merupakan keterangan – keterangan yang terkait pada sebuah entitas yang perlu disimpan dalam basis data. Atribut berfungsi sebagai penjelas pada sebuah entitas. Untuk menggambarkan atribut digunakan aturan sebagai berikut:
  - 1) Atribut digambarkan dengan simbol elips.
  - 2) Nama atribut dituliskan didalam simbol elips.
  - 3) Nama atribut merupakan kata benda, tunggal.
  - 4) Nama atribut sedapat mungkin menggunakan nama yang mudah dipahami dan dapat menyatakan maknanya dengan jelas.
3. Relasi merupakan hubungan antara sejumlah entitas yang berasal dari himpunan entitas yang berbeda. Aturan penggambaran relasi adalah sebagai berikut:
  - 1) Relasi dinyatakan dengan simbol belah ketupat.
  - 2) Nama relasi dituliskan didalam simbol belah ketupat.
  - 3) Nama relasi berupa kata kerja aktif.
  - 4) Nama relasi sedapat mungkin menggunakan nama yang mudah dipahami dan dapat menyatakan maknanya dengan jelas.

Komponen-komponen yang terlibat dalam *Entity Relationship Diagram* adalah sebagai berikut:

#### 1. **Objek Data (*entity*)**

Objek data adalah sekumpulan objek atau sesuatu yang dapat di bedakan atau didefinisikan secara unik. Objek data pada ERD disimbolkan dengan bentuk persegi panjang.

#### 2. ***Attribute***

*Attribute* adalah karakteristik dari entitas atau *relationship* yang menyediakan penjelasan detail tentang entitas.

### 3. *Relationship*

*Relationship* adalah hubungan yang terjadi antara satu *entity* atau lebih.

Ada tiga jenis *cardinality* dalam *relationship* yaitu:

1. *One to one* (satu ke satu).

Setiap entitas pada himpunan entitas A (pengojek) berhubungan dengan paling banyak satu entitas himpunan B (motor), dan begitu pula sebaliknya.

2. *One to many* (satu ke banyak).


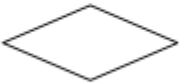

Setiap entitas pada himpunan entitas A (instruktur) berhubungan dengan banyak entitas pada himpunan entitas B (siswa), tetapi tidak sebaliknya. *Relationship* ini digambarkan sebagai berikut:

3. *Many to many* (banyak ke banyak).

Setiap entitas pada himpunan entitas A (siswa) berhubungan dengan banyak entitas pada himpunan entitas B (nilai), begitu pula sebaliknya. *Relationship* ini digambarkan sebagai berikut:

Adapun simbol-simbol yang digunakan pada *Entity Relationship Diagram* dapat dilihat pada table berikut ini.

**Tabel 2.4** Simbol *Entity Relationship Diagram (ERD)*

| Simbol  | Keterangan  |
|---|---|
|  | Melambangkan entitas yang mewakili objek sebenarnya di dunia nyata. |
|  | Menunjukkan kerelasian antara dua entitas atau lebih.               |
|  | Melambangkan atribut yang dimiliki oleh entitas.                    |

## 2.12. HTML

Menurut Jubilee Enterprise (2016:16) HTML adalah *Hypertext Markup Language* yang artinya adalah sebuah teks berbentuk link dan mungkin juga foto atau gambar – yang saat diklik, akan membawa si pengakses internet dari satu dokumen ke dokumen lainnya. Dalam praktiknya, *Hypertext* berwujud sebuah link yang bisa mengantar ke dunia internet yang sangat luas. Untuk membantu si pengakses berpindah dari satu tempat ke tempat lainnya, dibuatlah semacam dokumen yang nantinya akan disebut dengan istilah *website*. Untuk membuat *website*, kita membutuhkan *Markup*, yaitu tag (semacam kode) yang mengatur bagaimana *website* tersebut akan ditampilkan di jendela browser, seperti layout dan tampilan – tampilan visual yang biasa kita lihat didalam sebuah *website*. HTML adalah semacam bahasa yang ditunjukan oleh kata *Language* yang merupakan petunjuk bahwa HTML adalah semacam script pemrograman.

## 2.13. PHP

PHP (*Hypertext Preprocessor*) adalah bahasa *script* yang dapat ditanamkan atau disisipkan ke dalam HTML. PHP banyak dipakai untuk membuat program situs web dinamis. PHP sering juga digunakan untuk membangun sebuah CMS. PHP adalah bahasa pemrograman *script server-side* yang didesain untuk pengembangan web. Disebut bahasa pemrograman *server-side* karena PHP diproses pada komputer *server*. Hal ini berbeda dibandingkan dengan bahasa pemrograman *client-side* seperti JavaScript yang di proses pada web browser (*client*). PHP dapat digunakan dengan gratis dan bersifat *open source*. PHP dirilis dalam lisensi PHP License, sedikit berbeda dengan lisensi GNU *General Public License* (GPL) yang biasa digunakan untuk protek *open source*. Kemudahan dan kepopuleran PHP sudah menjadi standar bagi *programmer web* di seluruh dunia.

## 2.14. CSS (*Cascading Style Sheets*)

Menurut Juju Dominikus (2013:9) CSS (*Cascading Style Sheet*) secara sederhana adalah sebuah metode yang digunakan untuk mempersingkat penulisan HTML seperti *font*, *color*, *text* dan *tabel* menjadi lebih ringkas sehingga tidak terjadi pengulangan penulisan. CSS adalah bahasa *style sheet* yang digunakan

untuk mengatur tampilan dokumen. Dengan adanya CSS, memungkinkan kita untuk menampilkan halaman yang sama dengan format berbeda. CSS sendiri merupakan sebuah teknologi internet yang direkomendasi oleh *World Wide Web Consortium* atau W3C pada tahun 1996.

### 2.15. Bootstrap

Menurut Jubille Enterprise (2016:1) Bootstrap adalah *framework front-end* yang intuitif dan powerful untuk pengembangan aplikasi web yang lebih cepat dan mudah. Bootstrap menggunakan HTML, CSS dan JavaScript. Bootstrap dikembangkan oleh Mark Otto dan Jacob Thornton dari Twitter. *Framework* ini diluncurkan sebagai produk *open source* pada Agustus 2011 di GitHub. Bootstrap memiliki fitur – fitur komponen *interface* yang bagus seperti *Typography, Forms, Buttons, Tables, Navigation, Dropdowns, Alerts, Modals, Tabs, Accordion, Carousel* dan lain sebagainya.

### 2.16. Database

Menurut Wahana Komputer (2010:1) *Database* adalah sebuah struktur yang umumnya terbagi dalam 2 hal, yaitu sebuah *database* flat dan sebuah *database* relasional.

*Database* relasional lebih mudah dipahami dari pada *database* flat karena *database* relasional mempunyai bentuk yang sederhana serta mudah dilakukan operasi data. MySQL sendiri adalah sebuah *database* relasional. *Database* yang memiliki struktur relasional terdapat tabel – tabel untuk menyimpan data. Pada setiap tabel terdiri dari kolom dan baris serta sebuah kolom untuk mendefinisikan jenis informasi apa yang harus disimpan.

### 2.17. MySQL

Menurut Wahana Komputer (2010:5) MySQL adalah *Relational Database Management System* (RDBMS) dapat menangani data yang bervolume besar. Meskipun begitu, tidak menuntut *resource* yang besar. MySQL adalah *database* yang paling populer diantara *database – database* yang lain. MySQL adalah

program *database* yang mampu mengirim dan menerima data dengan sangat cepat dan *multi user*. MySQL memiliki dua bentuk lisensi, yaitu *free software* dan *shareware*.

### 2.18. **White Box Testing**

Menurut Bharat Bhushan Agarwal dan Sumit Prakash Tayal pada bukunya yang berjudul *Software Engineering* (2009:154) *White box testing* atau *Structural Testing* adalah pendekatan untuk pengujian di mana tes tersebut berasal dari pengetahuan tentang struktur dan implementasi perangkat lunak. *White box testing* biasanya diterapkan pada unit program yang relatif kecil seperti *subroutines* atau operasi yang terkait dengan objek. Sesuai namanya, tester dapat menganalisis kode dan menggunakan pengetahuan tentang struktur komponen untuk mendapatkan tes. Analisis kode dapat digunakan untuk menemukan berapa banyak kasus uji yang diperlukan untuk menjamin bahwa semua statemen dalam program dieksekusi setidaknya satu kali selama proses ini, tidak disarankan untuk merilis perangkat lunak yang berisi pernyataan yang belum diuji dan mungkin menjadi bencana. Tujuan ini tampaknya mudah tetapi tujuan sederhana dari pengujian struktural lebih sulit untuk dicapai dari pada yang mungkin muncul pada pandangan pertama.

### 2.19. **Forward Chaining**

Menurut B. Herawan Hatadi (2018:10) Metode *Forward Chaining* adalah teknik pencarian yang dimulai dengan fakta yang diketahui, kemudian mencocokkan fakta – fakta tersebut dengan bagian *IF* dari *rules IF\_THEN*. Bila ada fakta yang cocok dengan bagian *IF*. Maka *rules* tersebut dieksekusi. Bila sebuah *rules* dieksekusi, maka sebuah fakta baru (bagian *THEN*) ditambahkan ke dalam database. Setiap *rules* hanya boleh dieksekusi sekali saja.