BAB II

LANDASAN TEORI

2.1. Tinjauan Pustaka

Menurut Sani (2016:14) menyatakan bahwa, "Tinjauan pustaka adalah uraian tentang seluruh informasi terkait dengan topik-topik penelitian yang akan diuji".

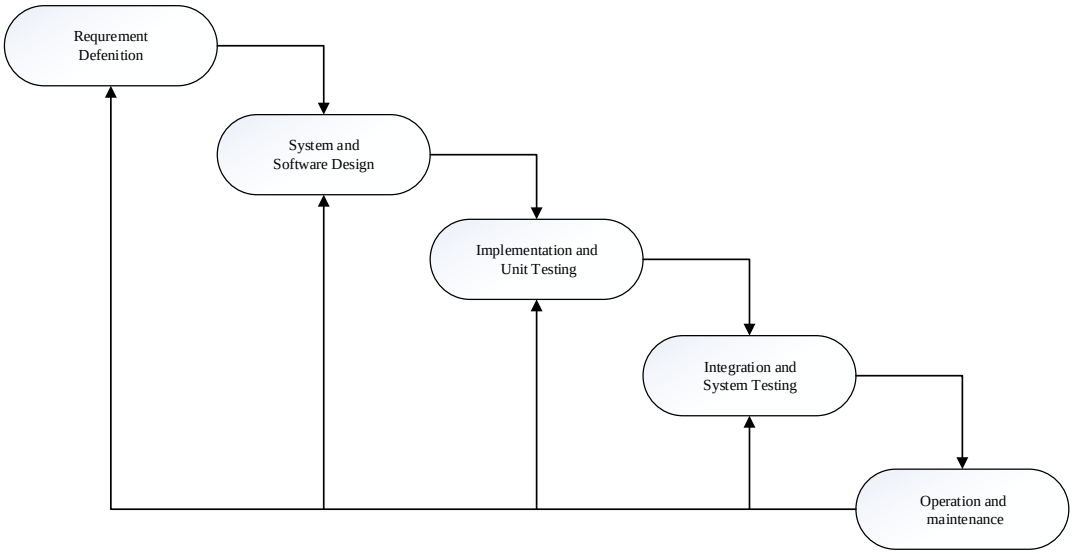
Menurut Zakiyudin (2011:9) menyatakan bahwa, "Sistem informasi adalah suatu sistem didalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi, mendukung operasi, dan bersifat manajerial dan kegiatan strategi dari suatu organisasi dan pihak luar tertentu dengan laporan yang diperlukan".

2.1.1. Konsep Dasar Sistem

Menurut Kusrini (2007:11) menyatakan bahwa, "Sistem adalah sebuah tatanan yang terdiri atas sejumlah komponen fungsional (dengan tugas/fungsi khusus) yang saling berhubungan dan secara bersama-sama bertujuan untuk memenuhi suatu proses/perkerjaan tertentu".

Menurut Muharto dan Arisandy Ambarita (2016:104) menyatakan bahwa, "Model *Waterfall* ini sebenarya adalah *"Linear Squential Model"*, yang sering juga disebut dengan *"classic life cycle"* atau model *waterfall*".

Menurut Pressman dan Sommerville dalam Muharto dan Arisandy Ambarita (2016b:105) menyatakan bahwa, "Metode ini terdiri dari beberapa step, seperti ditunjukan pada gambar berikut".



Sumber : Muharto dan Arisandy Ambarita (2016:105)

Gambar II.1.

Metode Waterfall

2.1.2. Konsep Dasar Pemrograman

1. *Adobe Dreamweaver*

   Menurut Hikmah dkk. (2015:2) menyatakan bahwa, "*Adobe Dreamweave*r merupakan salah satu *tool* yang banyak digunakan oleh pengembang web dikarenakan halamannya berbasis GUI (*Grafical User Interface*) sehingga memudahkan pengembang untuk mengembangkan website tidak terpaku konsep tekstual".

2. PHP (*Hypertext Preprocessor*)

   Menurut Anhar (2010:3) menyatakan bahwa, "PHP singkatan dari PHP: *Hypertext Preprocessor* yaitu bahasa pemrograman web server-side yang bersifat *open source*. PHP merupakan *script* yang terintegrasi dengan HTML dan berada pada *server* (*server side HTML embedded scripting*)".

3. *Web Browser*

Menurut Limantara (2009:1) menyatakan bahwa, "*Web Browser* adalah aplikasi perangkat lunak yang memungkinkan penggunanya untuk berinteraksi dengan teks, *image*, video, *games* dan informasi lainnya yang berlokasi pada halaman *web* pada *World Wide Web* (WWW) atau *Local Area Network* (LAN)".

4. HTML (*Hyper Text Markup Language*)

Menurut Faizal dan Irnawati (2015:1) menyatakan bahwa, "*Hyper Text Markup Language* (HTML) adalah bahasa *markah* yang digunakan untuk membuat sebuah halaman web, menampilkan berbagai informasi di dalam sebuah penjelajah web internet dan pemformatan hiperteks sederhana yang ditulis dalam berkas format ASCII agar dapat menghasilkan tampilan wujud yang terintegerasi".

5. MYSQL (*My Structure Query Language*)

Menurut Nugroho (2014:31) menyatakan bahwa, "MySQL adalah *software* atau program aplikasi *database*, yaitu software yang dapat kita pakai untuk menyimpan data berupa informasi teks dan juga angka".

6. XAMPP

Menurut Wicaksono (2008:7) menyatakan bahwa, "XAMPP adalah sebuah software yang berfungsi untuk menjalankan website berbasis PHP dan menggunakan pengolah data MySQL di komputer local."

7. PHP Myadmin

Menurut Rahman (2013:21) menyatakan bahwa, "PHP Myadmin adalah sebuah software berbasis pemrograman PHP yang dipergunakan sebagai administrator MySQL melalui browser (*web*) yang digunakan untuk managemen database".

8. *Database*

Menurut Arifin (2009:138) menyatakan bahwa, "*Database* atau basis data pada dasarnya adalah mengelola sekumpulan data sejenis. *Database* merupakan alat untuk menyimpan mengorganisasikan dan memeperoleh kembali informasi".

2.1.3. *Unifeld Modeling Language* (UML)

UML merupakan salah satu alat bantu yang sangat handal dalam bidang pengembangan sistem berorientasi objek karena UML menyediakan bahasa pemodelan visual yang memungkinkan pengembang sistem membuat blue print atas visinya dalam bentuk yang baku. UML berfungsi sebagai jembatan dalam mengkomunikasikan beberapa diagram. UML mempunyai banyak diagram yang dapat mengakomodasi berbagai sudut pandang dari suatu perangkat lunak yang akan dibangun. Diagram-diagram tersebut digunakan untuk (Henderi dkk, 2008:71):

1. Mengkomunikasikan ide
2. Melahirkan ide-ide baru dan peluang-peluang baru
3. Menguji ide dan membuat prediksi
4. Memahami struktur dan relasi-relasinya

Pemodelan menggunakan Unified Modeling Language merupakan metode pemodelan berorientasi objek dan berbasis visual. Karenanya pemodelan menggunakan UML merupakan pemodelan objek yang fokus pada pendefinisian struktur statis dan model sistem informasi yang dinamis daripada mendefinisikan data dan model proses yang tujuannya adalah pengembangan tradisional. UML menawarkan diagram yang dikelompokan menjadi lima perspektif berbeda untuk

memodelkan suatu sistem. Seperti satu set blue print yang digunakan untuk membangun sebuah rumah.

Berikut ini adalah penjelasan mengenai berbagai diagram UML :

1. *Use Case* Diagram

   Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. *Use case diagram* dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem.

2. *Class* Diagram

   Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).

   Class diagram menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. *Class* memiliki tiga area pokok :

a. Nama (dan *stereotype*).

b. Atribut.

c. Metoda.

Atribut dan metoda dapat memiliki salah satu sifat berikut :

a. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan.

b. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya.

c. *Public*, dapat dipanggil oleh siapa saja.

Class dapat merupakan implementasi dari sebuah *interface*, yaitu *class* abstrak yang hanya memilikimetoda. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class.* Dengan demikian *interface* mendukung resolusi metoda pada saat *run-time*.

Berikut merupakan hubungan (*interface*) antar *class*:

a. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukkan arah *query* antar *class*.

b. Agregasi, yaitu hubungan yang menyatakan bagian (“terdiri atas..”).

c. Pewarisan, yaitu hubungan hirarkis antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.

d. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di-*passing* dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence diagram* yang akan dijelaskan kemudian.

3. *Statechart* Diagram

   Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari *stimuli* yang diterima. Pada umumnya *statechart* diagram menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram*). Dalam UML, *state* digambarkan berbentuk segiempat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu. Transisi antar *state* umumnya memiliki kondisi *guard* yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam siku. *Action* yang dilakukan sebagai akibat dari *event* tertentu dituliskan dengan diawali garis miring. Titik awal dan akhir digambarkan berbentuk lingkaran berwarna penuh dan berwarna setengah.

4. *Activity* Diagram

   Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity* diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity* diagram merupakan *state* diagram khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-

jalur aktivitas dari level atas secara umum. Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas. Sama seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan behaviour pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal. *Activity* diagram dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.

5. *Sequence* Diagram

   Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence* diagram terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence* diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan. Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah message. Untuk objek-objek yang memiliki sifat

khusus, standar UML mendefinisikan *icon* khusus untuk objek *boundary*, *controller* dan *persistent entity*.

6. *Collaboration* Diagram

   Collaboration diagram juga menggambarkan interaksi antar objek seperti *sequence* diagram, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian *message*. Setiap *message* memiliki *sequence number*, di mana *message* dari level tertinggi memiliki nomor 1. *Messages* dari level yang sama memiliki prefiks yang sama.

7. *Component* Diagram

   Component diagram menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency*) di antaranya. Komponen piranti lunak adalah modul berisi *code*, baik berisi *source code* maupun *binary code*, baik *library* maupun *executable*, baik yang muncul pada *compile time*, *link time*, maupun *run time*. Umumnya komponen terbentuk dari beberapa *class* dan/atau *package*, tapi dapat juga dari komponen-komponen yang lebih kecil. Komponen dapat juga berupa *interface*, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.

8. *Deployment* Diagram

   Deployment/physical diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal. Sebuah *node* adalah server, *workstation*, atau piranti keras lain yang digunakan untuk men-

deploy komponen dalam lingkungan sebenarnya. Hubungan antar *node* (misalnya TCP/IP) dan *requirement* dapat juga didefinisikan dalam diagram ini.

2.1.4. *Entity Relationship Diagram* (ERD)

Menurut Kusrini (2007:21) menyatakan bahwa, "Perancangan basis data dengan menggunakan model entity relationship adalah dengan menggunakan *Entity Relationship Diagram* (ERD). Terdapat tiga notasi dasar yang bekerja pada model E-R yaitu : *entity sets, relationship set, attributes* ".

Menurut Sutanta (2011:91) dalam bukunya yang berjudul "Basis Data Dalam Tinjauan Konseptual" menjelaskan bahwa "Entity Relationship Diagram (ERD) merupakan suatu model data yang dikembangkan berdasarkan objek."

Dari beberapa pengertian diatas, dapat disimpulkan bahwa ERD merupakan gambaran grafis dari suatu model data yang menyertakan deskripsi detail dari seluruh entitas (*entity*), hubungan (*relationship*), dan batasan (*constraint*) untuk memenuhi kebutuhan sistem analis dalam menyelesaikan pengembangan sebuah sistem.

Komponen Entity Relationship Diagram adalah sebagai berikut :

1. Entitas merupakan suatu objek yang dapat dibedakan dari lainnya yang dapat diwujudkan dalam basis data. Objek dasar dapat berupa orang, benda, atau hal yang keterangannya perlu disimpan didalam basis data. Untuk menggambarkan sebuah entitas digunakan aturan sebagai berikut:

a. Entitas dinyatakan dengan simbol persegi panjang.

b. Nama entitas dituliskan didalam simbol persegi panjang.

c. Nama entitas berupa kata benda, tunggal.

d. Nama entitas sedapat mungkin menggunakan nama yang mudah dipahami dan dapat menyatakan maknanya dengan jelas.

2. Atribut 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:

   a. Atribut digambarkan dengan simbol ellips.

   b. Nama atribut dituliskan didalam simbol ellips.

   c. Nama atribut merupakan kata benda, tunggal.

   d. Nama atribut sedapat mungkin menggunakan nama yang mudah dipahami dan dapat menyatakan maknanya dengan jelas.

3. Relasi Relasi merupakan hubungan antara sejumlah entitas yang berasal dari himpunan entitas yang berbeda. Aturan penggambaran relasi adalah sebagai berikut :

   a. Relasi dinyatakan dengan simbol belah ketupat.

   b. Nama relasi dituliskan didalam simbol belah ketupat.

   c. Nama relasi berupa kata kerja aktif.

   d. Nama relasi sedapat mungkin menggunakan nama yang mudah dipahami dan dapat menyatakan maknanya dengan jelas.

4. *Indicator Type*

   a. *Indicator type associative object* berfungsi sebagai suatu objek dan suatu

relationship.

b. *Idicator type supertype* terdiri dari suatu objek dan satu subkategori atau lebih yang dihubungkan dengan satu *relationship* yang tidak bernama.

5. *Cardinality Ratio* atau *Mapping Cardinality*

Cardinality Ratio adalah menjelaskan hubungan batasan jumlah keterhubungan satu *entity* dengan *entity* yang lainnya atau banyaknya *entity* yang bersesuaian dengan *entity* yang lain melalui *relationship*. Jenis *cardionality ratio* sebagai berikut:

a. One to one (1:1) adalah hubungan satu *entity* dengan satu *entity*.

b. *One to many* (M:N) atau *Many to one* (M:1) adalah hubungan suatu *entity* dengan banyak *entity* atau banyak *entity* dengan satu *entity*.

c. *Many to many* (M:N) adalah hubungan banyak *entity* dengan banyak *entity*

6. *Derajat Relationship*

Derajat Relationship menyatakan jumlah *entity* yang berpatisipasi didalam suatu *relationship*.

a. *Unary degree (*derajat satu) adalah derajat yang dimiliki satu *relationship* untuk dua buah *entity*.

b. *Binary degree (*derajat dua) adalah derajat yang memiliki satu *relationship* untuk dua buah *entity*.

c. *Tenary degree* (derajat tiga) adalah derajat yang memiliki satu *relationship* untuk tiga atau lebih *entity*.

7. *Participation Contrain*

Menjelaskan apakah keberadaan suatu *entity* tergantung pada pada hubungannya dengan *entity* lain. Terdapat dua macam *participation constraints* yaitu :

a. *Total participation*, yaitu keberadaan suatu entity tergantung pada hubungannya dengan entity lain dalam E-R digambarkan dua garis penghubung antar entity dan relationship.
b. *Partial participation*, yaitu keberadaan suatu *entity* tidak tergantung pada hubungan dengan *entity* lain. Di dalam E-R digambarkan dengan satu garis penghubung.

8. Representasi danri *entity set*

Entity set dipresentasikan dalam bentuk table dan nama yang unik. Setiap table terdiri dari sejumlah kolom. Masing-masing kolom diberi nama yang unik pula. *Entity set* terbagi atas:

a. *Strong entity set*, *entity set* yang satu atau lebih atributnya digunakan oleh *entity set* lain sebagai *key* digunakan dengan empat persegi panjang.
b. *Weak entity set*, *entity set* yang tergantung terhadap *strong entity set.* Keberadaan *entity* tersebut tergantung keberadaan *entity* lain. *Entity* lain tersebut disebut *identifying owner* dan *relationship* disebut *identifying relationship*. *Weak entity* selalu mempunyai total *participation constraint* dengan *identifying owner*. *Weak entity* digambarkan dengan persegi bertumpuk.

Langkah-langkah membuat E-R Diagram sebagai berikut:

1) Tentukan *entity-entity* yang diperlukan.
2) Tentukan *relationship* antar *entity-entity*.
3) Tentukan *cardinality ratio* dan *participation constraint*.

4) Tentukan *attribute-attribute* yang diperlukan dari tiap entity.

5) Tentukan key diantara *attribute-attribute.*

6) Hindari penamaan *entity*, *relationship*, dan *attribute* yang sama.

2.2. Penelitian Terkait

Menurut Suhendar (2015:1) menyimpulkan bahwa:
Salah satu upaya untuk memperbaiki proses PSB dan meningkatkan pelayanan kepada calon siswa, maka dibutuhkan sebuah sistem yang dapat melakukan proses PSB yang tidak dibatasi oleh tempat. Salah satu teknologi yang dapat menyelesaikan permasalahan itu adalah dengan memanfaatkan teknologi *website*. Aplikasi PSB berbasis *website P*ini diharapkan dapat membantu calonsiswa barudalam mendapatkan informasi. Sehingga calon siswa akan mendapatkan informasi pendaftaran dan hasil seleksi pendaftaran tanpa harus datang ke sekolah.

Menurut Putra, Ferdian Sukarno dkk (2011:35) menyatakan bahwa:
Dalam dunia pendidikan, terutama pendidikan tinggi akan membutuhkan informasi untuk merampingkan dan menyederhanakan proses akademik di dalamnya. Manajemen dan pengolahan data masih dilakukan secara manual, kadang-kadang dapat menyebabkan kesalahan data yang disebabkan oleh faktor kesalahan manusia.Oleh karena itu,perlu bahwa system informasi mampu mengelola dan mengolah data secara efektif dan efisien sehingga data menjadi lebih akurat.