BAB II
LANDASAN TEORI

2.1 Pengertian Sistem

Sistem berasal dari bahasa Latin (*systēma*) dan bahasa Yunani (*sustēma*) adalah suatu kesatuan yang terdiri komponen atau elemen yang dihubungkan bersama untuk memudahkan aliran informasi, materi atau energi. Sistem merupakan kumpulan bagian-bagian atau sub-sub sistem yang disatukan dan dirancang untuk mencapai suatu tujuan.

Sistem adalah suatu kesatuan yang terdiri atas komponen atau elemen yang saling berinteraksi, saling terkait atau saling bergantung membentuk keseluruhan yang komplek. Tujuan dari sistem tersebut adalah untuk mengorganisasikan sistem informasi yang baru agar dapat mengatasi berbagai masalah yang terjadi pada suatu organisasi, serta memberikan pengertian mengenai suatu bentuk sistem yang ada pada suatu organisasi serta trik-trik manageman yang berkaitan dengan sistem informasi manageman (SIM) berbasis komputer.

Analisa sistim sistem adalah suatu studi dari sistem yang telah ada dengan tujuan untuk merancang sistem yang baru atau memperbaiki kekurangan dari sistem yang telah ada (Mc Leod). Analisa sistem adalah kegiatan menemukan atau mengidentifikasikan masalah, mengevaluasi, membuat model serta membuat spesifikasi sistem (Pressman).

2.1.1 Elemen-elemen Sistem

Suatu system mempunyai elemen-elemen tertentu yaitu : sistem terbuka dan sistem tertutup.

a. Sistem Terbuka adalah sistem yang berhubungan dengan lingkungannya melalui arus sumber daya. Dimana sumber daya mengalir dari elemen input, melalui Elemen Transformasi, kepada Elemen Output, suatu mekanisme kontrol memantau proses transformasi untuk meyakinkan bahwa system tersebut memenuhi tujuannya. Mekanisme kontrol ini dihubungkan pada

arus sumber daya dengan memakai suatu lingkaran umpan balik *(feedback loop)* yang mendapatkan informasi dari output system dan menyediakan informasi dari mekanisme kontrol. Mekanisme kontrol membandingkan sinyal-sinyal umpan balik dengan tujuan, dan mengarahkan sinyal pada elemen input jika sistem operasi memang perlu dirubah.

b. Sistem tertutup adalah sistem yang tidak berhubungan dengan lingkungannya.

2.1.2 Tingkatan Sistem

a. Supra sistem : sistem yang lebih besar atau jika suatu system adalah bagian

dari sistem lain yang lebih besar.

b. System : susunan subsistem-subsistem yang mempunyai tujuan tertentu.

c. Subsistem : Bagian dari sistem dan merupakan unsur terkecil sistem.

2.1.3 Karakteristik Sistem

Sistem mempunyai beberapa karakteristik atau sifat tertentu, (Jogiyanto,1993:3) antara lain :

a. Komponen sistem (*Component*) : Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang saling bekerja sama membentuk suatu komponen sistem. Setiap subsistem mempunyai sifat-sifat dari sistem untuk menjalankan suatu fungsi tertentu dan mempengaruhi proses sistem secara keseluruhan.

b. Batasan sistem (*Boundary*) : Merupakan daerah yang membatasi suatu sistem

dengan sistem yang lain atau dengan lingkungan kerjanya. Batas sistem ini memungkinkan suatu sistem dipandang sebagai satu kesatuan. Batas sistem menunjukkan ruang sistem (*scope)* dari sistem tersebut.

c. Sub sistem (*Sub system*) : Bagian-bagian dari sistem yang beraktivitas dan berinteraksi satu sama lain untuk mencapai tujuan dengan sasarannya masing-masing.

d. Lingkungan luar sistem (*Environment*) : Suatu sistem yang ada di luar dari batas sistem yang dipengaruhi oleh operasi sistem. Lingkungan luar sistem dapat bersifat menguntungkan dan dapat juga merugikan sistem tersebut.

e. Penghubung sistem (*Interface*) : Media penghubung antara suatu sub sistem dengan sub sistem lain. Adanya penghubung ini memungkinkan berbagai sumber daya mengalir dari suatu sub sistem ke subsistem lainnya. Keluaran *(output)* dari suatu subsistem akan menjadi masukan *(input)* untuk subsistem yang lainnya dengan melalui penghubung. Dengan penghubung satu subsistem dapat berinteraksi dengan subsistem lainnya membentuk satu kesatuan.

f. Masukan sistem (*Input*) : Energi yang masuk ke dalam sistem, berupa perawatan dan sinyal. Masukan perawatan adalah energi yang di masukkan supaya sistem tersebut dapat berinteraksi.

g. Keluaran sistem (*Output*) : Hasil yang diolah dan diklasifikasi menjadi keluaran yang berguna dan sisa pembuangan. Keluaran dapat merupakan masukkan untuk subsistem yang lain.

h. Pengolahan sistem (*Process*) : Suatu sistem dapat mempunyai suatu bagian pengolahan yang akan mengubah masukan menjadi keluaran.

i. Sasaran sistem (*Object*) : Suatu sistem pasti mempunyai tujuan *(goal)* atau sasaran. Kalau suatu sistem tidak mempunyai sasarn, maka operasi tidak akan gunanya. Sasaran dari sistem sangat menentukan sekali masukkan yang dibutuhkan sistem dan keluaran yang akan dihasilkan sistem. Suatu sistem dikatakan berhasil bila mengenai sasaran atau tujuannya.

2.1.4 Pengertian Informasi

Informasi adalah data yang telah diproses dan sebagaimana layaknya sumber daya yang lain (orang, mesin) yang dapat dikelola. Kualitas dari suatu informasi tergantung pada tiga hal utama antara lain : (Jogiyanto:37)

a. Akurat (*accuracy*)

   Informasi harus bebas dari kesalahan-kesalahan dan tidak bias atau menyesatkan, dan harus jelas mencerminkan maksudnya. Ketidakakuratan dapat terjadi karena sumber informasi (data) mengalami gangguan atau kesengajaan sehingga merusak atau merubah data-data asli tersebut.

b. Tepat waktu (*timelines*)

   Informasi yang dihasilakn atau dibutuhkan tidak boleh terlambat (usang). Informasi yang usang tidak mempunyai nilai yang baik, sehingga kalau digunakan sebagai dasar dalam pengambilan keputusan akan berakibat fatal atau kesalahan dalam keputusan dan tindakan. Kondisi demikian menyebabkan mahalnya nilai suatu informasi, sehingga kecepatan untuk mendapatkan, mengolah dan mengirimkannya memerlukan teknologi-teknologi yang baru.

c. Relevan (*Relevancy*)

   Berarti informasi harus memberikan manfaat bagi pemakainya. Relevansi informasi untuk tiap-tiap orang dengan yang lain berbeda.

2.1.5 Pengertian Sistem Informasi

Menurut Kadir(2003:10) Sistem informasi mencakup sejumlah komponen (manusia, komputer, teknologi informasi, dan prosedur kerja), ada sesuatu yang diproses (data menjadi informasi), dan dimaksudkan untuk mencapai suatu sasaran atau tujuan.

Menurut Jogiyanto (2005:11) sistem informasi adalah suatu sistem didalam suatu organisasi yang mempertemukan kebutuhan pengelolahan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu

organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang dikeluarkan.

Dari kedua teori diatas dapat disimpulkan bahwa sistem informasi yaitu: kegiatan atau aktifitas yang melibatkan serangkaian proses, berisi **informasi-informasi** yang digunakan atau diorganisasikan untuk mencapai tujuan dalam sebuah organisasi

2.2 Analisa dan Perancangan Sistem Berorientasi Obyek Dengan UML

UML adalah sebuah "bahasa pemodelan" yang menspesifikasikan, menvisualisasikan, membangun dan mendokumentasikan kerangka dari sebuah sistem *software*.

Menurut pencetusnya James Rumbaugh, Ivar Jacobson, and Grady Booch (1999 : 119-120), *UML* didefinisikan sebagai *"bahasa visual untuk menjelaskan, memberikan spesifikasi, merancang, membuat model, dan mendokumentasikan aspek-aspek dari sebuah sistem"*.

UML merupakan penerus dari gelombang metode perancangan dan analisa berorientasi objek (*object-oriented analysis and design metode*) yang berkembang pada era 80-an sampai 90-an. Pada masa itu, banyak metode berorientasi objek yang dikembangkan antara lain : *Booch Cold Yourdon, Fusion,* OMT *(Object Modeling Technique), OOSE, Shlaer-Mellor, Martin-Odell,* dan sebagainya.

Ide UML sendiri bermula dari keinginan Grady Booch (Booch) untuk membuat sebuah metode bersama untuk unifikasi. Pada tahun 1994 James Rumbaugh (UMT) bergagung bersama Booch di perusahaan Rational kemudian mereka menghasilkan sebuah metode yang disebut dengan *Unified Methode* dan dirilis untuk pertama kali pada bulan Oktober 1995 dengan versi 0.8. *Unified Methode* menjadi sangat populer dan banyak dibicarakan serta dijadikan notasi untuk berbagai makalah.

Pada musim gugur 1995, Ivar Jacobson bergabung dengan perusahaan Rational. Dengan menggunakan *Use Case* dan model interaksi antar objek, mereduksi kekurangan dan sebelumnya serta membawa ide baru terhadap *Unified Methode*, dan kemudian barulah *Unified Mothode* berganti nama menjadi *Unified Methode Language* (UML) versi 0.9 dan 0.91 dirilis pada bulan Juni dan Oktober 1996. Tahun 1997 UML versi 1.1 muncul dan saat ini versi terbaru adalah 1.5 yang dirilis bulan Maret 2003. Sejak itulah UML menjelma menjadi standar bahasa pemodelan unutk aplikasi berorientasi objek.

2.2.1 Analisa Sistem Berorientasi Obyek

2.2.1.1 Activity Diagram

Activity diagram menggambarkan proses bisnis dan urutan aktifitas dalam sebuah proses, yang mana dipakai pada *business modelling* untuk memperlihatkan urutan aktifitas proses bisnis karena bermanfaat untuk membantu memahami proses secara keseluruhan dalam memodelkan sebuah proses.

Dengan kata lain, *activity diagram* adalah tehnik untuk mendeskripsikan logika prosedural, proses bisnis dan aliran kerja dalam banyak kasus. *Activity diagram* mempunyai peran seperti halnya *flowchart,* akan tetapi perbedaannya dengan *flowchart* adalah *activity diagram* bisa mendukung perilaku paralel sedangkan *flowchart* tidak bisa.

Activity diagram dibuat berdasarkan sebuah atau beberapa *use case* pada *use case diagram,* atau bahkan tanpa menggunakan *use case diagram.* Sebuah *activity diagram* mempunyai :

1) *Start point (initial node),* dimana diletakkan pada pojok kiri atas.
2) *End point (activity final node)*
3) *activities,* dimana menggambarkan proses bisnis dan dikenal sebagai *activity state.*

Jenis-jenis *activity* :

a. *Black hole activities*

Ada masukan dan tidak ada keluaran, biasanya digunakan bila dikehendaki ada 1 atau lebih transisi.

b. *Miracle activities*

Tidak ada masukan tetapi ada keluarannya, biasanya dipakai pada waktu *start point* dan dikehendaki ada 1 atau lebih transisi.

c. *parallel activities*

Suatu activity yang berjalan secara berbarengan. Terdiri dari:

Fork (percabangan)

Mempunyai 1 transisi masuk dan 2 atau lebih transisi keluar.

Ketika ada > 1 transisi masuk ke fork yang sama, gabunglah dengan sebuah *decision point.*

a. *Join* (penggabungan)

Mempunyai 2 atau lebih transisi masuk dan hanya 1 transisi keluar.

b. *Decision point*

Digambarkan dengan lambang wajik atau belah ketupat. Mempunyai transisi (sebuah garis dari/ke *decision point*). Setiap transisi yang ada harus mempunyai *GUARD* (kunci). Tidak ada sebuah keterangan (pertanyaan) pada tengah belah ketupat seperti pada *flowchart.*

c. *Guard* (kunci)

Adalah sebuah kondisi benar sewaktu melewati sebuah transisi. Digambarkan dengan diletakkan diantara tanda [ ]. Tanda [*otherwise*] guard untuk menangkap suatu kondisi yang belum terdeteksi. Setiap transisi dari/ke *decision point* harus mempunyai *guard* yang harus konsisten dan lengkap serta tidak *overlap.*

d. *Swimlane*

Sebuah cara untuk mengelompokkan *activity* berdasarkan *actor* (mengelompokkan *activity* dalam sebuah urutan yang sama). *Actor* bisa ditulis nama *actor* ataupun sekaligus dalam lambang actor (*stick figure*)

pada *use case diagram*. Swimlane digambar secara *vertical*, walaupun kadang-kadang digambar secara *horizontal*.

e. *Swimarea*

Ketika sebuah *activity diagram* mempunyai banyak *swimlane*, perlu dipikirkan dengan pendekatan *swimarea*. *Swimarea* mengelompokkan *activity* berdasarkan kegiatan didalam *use case*.

2.2.1.2 Analisa Dokumen Keluaran

Adalah sistem analisa mengenai keluaran-keluaran yang dihasilkan dari sebuah sistem.

2.2.1.3 Analisa Dokumen Masukan

Analisa masukan adalah bagian dari pengumpulan informasi tentang sistem yang sedang berjalan. Tujuan analisa masukan adalah memahami prosedur berjalan.

2.2.1.4 Use Case Diagram

Use case diagram menggambarkan kebutuhan sistem dari sudut pandang user dan memfokuskan pada proses komputerisasi. Sebuah *use case* dapat menggambarkan hubungan antara *use case* dengan *actor*. Secara umum *use case* adalah pola perilaku sistem dan urutan transaksi yang berhubungan yang dilakukan oleh satu *actor*.

Use case diagram terdiri dari :

1) *Use case*

Use Case adalah *deskripsi* fungsi dari sebuah sistem dari perspektif pengguna. *Use case* dibuat berdasarkan keperluan *actor*, merupakan 'apa' yang dikerjakan sistem, bukan 'bagaimana' sistem mengerjakannya. *Use case* diberi nama yang menyatakan apa hal yang dicapai dari hasil interaksinya dengan *actor* dan dinotasikan dengan gambar (*horizontal ellipse*).

Use case biasanya menggunakan kata kerja dan sebuah nama *use case* boleh terdiri dari beberapa kata dan tidak boleh ada 2 *use case* yang memiliki nama yang sama. *Use case* diagram tidak terpengaruh urutan waktu, meskipun demikian supaya mudah dibaca perlu penyusunan *use case*.

2) *Actor*

Actor menggambarkan orang, sistem atau *eksternal entitas/stakeholder* yang menyediakan atau menerima informasi dari sistem. *Actor* adalah *entity eksternal* yang berhubungan dengan sistem yang berpartisipasi dalam *usecase*.

Actor dilukiskan dengan peran yang mereka mainkan dalam *use case*, seperti Pelanggan, kasir, dan lain-lain. Simbol actor didalam *UML* digambarkan sebagai berikut :

3) *Associations*

Associations menggambarkan bagaimana *actor* terlibat dalam *use case* dan bukan menggambarkan aliran data atau informasi. *Association* digambarkan dengan sebuah garis berpanah terbuka pada salah satu ujungnya yang menunjukkan arah relasi.

Jenis-jenis relasi yang bisa timbul pada *use case* diagram adalah sebagai berikut

a. *Association* antara *actor* dan *use case*

Ujung panah pada *association* antara *actor* dan *use case* mengindikasikan siapa/apa yang meminta interaksi dan bukannya mengindikasikan aliran data.*Association* antara *actor* dan *use case* sebaiknya menggunakan garis tanpa panah. *Association* antar *actor* dan *use case* yang menggunakan panah terbuka untuk mengindikasikan bila *actor* berinteraksi secara pasif dengan sistem.

b. *Association* antara *use case*

Digunakan ketika dalam penulisan *use case-use case* yang berbeda-beda terdapat deskripsi-deskripsi yang sama, maka relasi ini dapat digunakan untuk menghindari penulisan deskripsi yang berulang-ulang.

Sebuah *use case* dapat meng-*include fungsionalitas use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include dieksekusi* secara normal. Sebuah *use case* dapat *di-include* oleh lebih dari satu *use case* lain, sehingga duplikasi *difungsionalitas* dapat dihindari dengan cara menarik keluar *fungsionalitas* yang *common.*

Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviournya* sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

2.2.1.5 Deskripsi Use Case Diagram

Membuat diagram *use case* adalah proses dua langkah: pertama, pengguna bekerja dengan tim proyek untuk menulis *deskripsi use case* berbasis teks, dan kedua tim proyek menerjemahkan kasus *use case description* ke dalam *diagram use case* formal, baik *deskripsi use case* dan *use case diagram* ini didasarkan pada persyaratan diidentifikasi dan *activity diagram* deskripsi program bisnis. *Deskripsi use case* berisi semua informasi yang dibutuhkan untuk menghasilkan *diagram use case.* Meskipun dimungkinkan untuk melompati langkah menggunakan *deskripsi use case* dan bergerak langsung untuk membuat diagram use case dan diagram lainnya yang sesuai, pengguna seringkali memiliki kesulitan menggambarkan proses bisnis, mereka hanya menggunakan diagram *use case*.

Melalui penciptaan *deskripsi use case*, pengguna dapat menggambarkan rincian yang dibutuhkan dari masing-masing *use case* individu. Untuk yang akan datang *deskripsi use case* pertama gunakan atau menggunakan diagram *use case* secara teknis, itu benar-benar tidak masalah. Keduanya harus dilakukan untuk sepenuhnya menggambarkan persyaratan bahwa sistem informasi harus bertemu.

Deskripsi use case berisi semua informasi yang dibutuhkan untuk membangun diagram yang mengikuti, tetapi menyatakan dalam cara yang lebih formal yang biasanya sederhana bagi pengguna untuk mengerti. Ada tiga bagian dasar untuk suatu *deskripsi use case*: *Overview Information, Relationship, Flow of Events.*

Overview Information : Informasi gambaran mengidentifikasi *use case* dan menyediakan informasi didominasi dasar tentang *use case*. Nama *use case* dari *use case* harus menjadi frase kata kerja-kata benda (misalnya, Membuat Penunjukan). Nomor ID *use case* menyediakan cara yang unik untuk menemukan setiap *use case* dan juga memungkinkan tim untuk menelusuri keputusan desain kembali ke persyaratan spesifik. Seperti telah disebutkan, jenis *use case* adalah tinjauan atau detail dan penting atau nyata. *Actor* utama biasanya pemicu *use case*-orang atau hal yang dimulai pelaksanaan *use case*. Tujuan utama dari *use case* adalah untuk memenuhi tujuan dari *actor* utama. Gambaran singkat biasanya satu kalimat yang menjelaskan inti dari *use case.*

Relationships : Use case relationships hubungan *use case* menjelaskan bagaimana kasus penggunaan terkait dengan *use case* lainnya dan pengguna. Ada empat tipe dasar hubungan: *association, extend, include, and generalization.* Sebuah dokumen hubungan asosiasi komunikasi yang terjadi antara *use case* dan aktor-aktor yang menggunakan *use case.* Seorang aktor adalah representasi UML untuk peran bahwa seorang pengguna bermain di *use case*. Semua aktor yang terlibat dalam *use case* didokumentasikan dengan hubungan asosiasi. Hubungan *extend* merupakan perpanjangan fungsional *use case* untuk menggabungkan perilaku opsional. Hubungan sertakan merupakan dimasukkannya wajib *use case* lain. Termasuk hubungan memungkinkan dekomposisi-fungsional putus dari beberapa *use case* yang kompleks menjadi sederhana. Generalisasi yang memungkinkan kasus gunakan untuk mendukung warisan.

2.2.2 Perancangan Sistem Berorientasi Obyek

Perancangan sistem Berorientasi Objek merupakan tahap lanjutan setelah analisa berorientasi objek.

"*Perancangan berorientasi objek adalah suatu pendekatan yang digunakan untuk menspesifikasikan kebutuhan – kebutuhan sistem dengan mengkolaborasikan objek – objek, atribut – atribut, method – method yang ada.*"(Jeffery L., Whitten et al, 2004 : 686)

Merupakan proses spesifikasi yang terperinci atau pendefinisian dari kebutuhan-kebutuhan fungsional yang menggambarkan bagaimana suatu sistem itu dibentuk. Perancangan sistem berorientasi objek ditujukan untuk mensistematis proses pendesainan dan menghasilkan pendesainan model program. Serta memberikan gambaran pemecahan masukan dengan efektif.

2.2.2.1 ERD

ERD merupakan suatu model untuk menjelaskan hubungan antar data dalam basis data berdasarkan objek-objek dasar data yang mempunyai hubungan antar relasi.

ERD memodelkan struktur data dan hubungan antar data, untuk menggambarkannya digunakan beberapa notasi dan simbol.

Pada dasarnya ada tiga simbol yang digunakan, yaitu :

a. **Entity**

Entity merupakan objek yang mewakili sesuatu yang nyata dan dapat dibedakan dari sesuatu yang lain (Fathansyah, 1999: 30). Simbol dari *entity* ini

biasanya digambarkan dengan persegi panjang.

b. **Atribut**

Setiap entitas pasti mempunyai elemen yang disebut *atribut* yang berfungsi untuk mendeskripsikan karakteristik dari entitas tersebut. Isi dari atribut mempunyai sesuatu yang dapat mengidentifikasikan isi elemen satu dengan

yang lain. Gambar *atribut* diwakili oleh simbol elips.

c. **Hubungan / Relasi**

Hubungan antara sejumlah entitas yang berasal dari himpunan entitas yang berbeda.

2.2.2.2 Logical Record Struktur (LRS)

LRS adalah suatu terstruktur yang terdiri dari sejumlah *record type*, dimana setiap *record type* dinyatakan dalam bentuk kotak persegi panjang dan memiliki sebuah nama yang unik ditulis diluar kotak dan nama *field* yang ditulis didalam kotak yang berisi link diantara *record type*, dimana setiap *link* diberi label dengan *field* yang muncul pada kedua buah *record* yang dihubungkan oleh *link* tersebut.

Konversi ER-Diagram ke *Logical Record Structure* dan Relasi (LRS) ER-Diagram harus di ubah ke bentuk LRS (struktur *record* secara logik). Dari bentuk *Logical Record Structure* terdiri dari *link-link* diantara tipe *record. Link* ini menunjukkan arah dari satu tipe *record* lainnya. Banyak *link* dari LRS yang diberi tanda *field-field* yang kelihatan pada kedua *link type record.* Penggambaran LRS mulai dengan penggambaran model yang dimengerti. Dua metode yang dapat digunakan, dimulai dengan hubungan kedua model yang dapat dikonversikan ke LRS. Metode yang lain dimulai dengan ER Diagram yang langsung dikonversikan ke LRS.

Logical record structure inilah yang nantinya dapat ditransformasikan ke bentuk relasi (tabel).

2.2.2.3 Tabel/Relasi

Tabel adalah bentuk pernyataan data secara grafis 2 (dua) dimensi, yang terdiri dari kolom dan baris. Relasi adalah bentuk *visual* dari sebuah *file*, dan tiap *tuple* dalam sebuah *field*, atau yang dalam bentuk lingkaran. Diagram E-R dikenal dengan sebutan atribut. Konversi dari *logical record structure* dilakukan dengan cara:

a) Nama *logical record structure* menjadi nama relasi

b) Tiap atribut menjadi sebuah kolom didalam relasi.

2.2.2.4 Spesifikasi Basis Data

Basis data adalah kumpulan data (arsip atau *file*) yang saling berhubungan yang disimpan dalam media penyimpanan elektronis agar dapat dimanfaatkan kembali dangan cepat dan mudah.

Sedangkan sistem basis data adalah kumpulan file atau tabel yang saling berhubungan yang memungkinkan beberapa pemakai dan / atau program lain untuk mengakses dan memanipulasi *file-file* (tabel) tersebut.

2.2.2.5 Rancangan Dokumen Keluaran

Rancangan keluaran merupakan informasi yang akan dihasilkan dari keluaran sistem yang dirancang.

2.2.2.6 Rancangan Dokumen Masukan

Rancangan masukan merupakan data yang dibutuhkan untuk menjadi masukan sistem yang dirancang.

2.2.2.7 Rancangan Layar Program

Rancangan masukan merupakan data yang dibutuhkan untuk menjadi masukan sistem yang dirancang.

2.2.2.8 Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam maupun di sekitar sistem (temasuk pengguna, *display* dan sebagainya) berupa *message* yang digambarkan terhadap waktu.

Dan biasa digunakan untuk menggambarkan skenario atau langkah-langkah yang dilakukan sebagai *respon* dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang menggerakkan aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara i*nternal* dan *output* apa yang dihasilkan.

Sequence diagram memiliki :

a. *Actor*, Menggambarkan orang yang sedang berinteraksi dengan sistem.
b. *Entity Object*, suatu objek yang berisi informasi kegiatan yang terkait yang tetap

   dan disimpan ke dalam suatu *database*. (Jeffery L. Whitten et al, 2004: 686)
c. *Interface/Boundary Object*, sebuah objek yang menjadi penghubung antara *user*

   dengan sistem. Contohnya *window*, *dialogue box* atau *screen*(tampilan layar).

   (Jeffery L. Whitten et al, 2004 : 686)
d. *Simple Message*, simbol pengiriman pesan dari sebuah objek ke objek lain. (Jeffery L. Whitten et al, 2004 : 704)
e. *Lifeline*, garis titik - titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*. (Munawar, 2005 : 87;89).

2.2.2.9 *Class Diagram (Entity Class)*

Class diagram merupakan diagram paling umum dipakai disemua pemodelan disemua orientasi objek. Pemodelan *class* merupakan pemodelan paling utama dipendekatan berorientasi objek. Pemodelan *class* menunjukkan *clas-clas* yang ada di sistem dan hubungan antar *class*. *Class diagram* digambarkan dengan sebuah kotak dengan 3 *section*.

Komponen-komponen *class diagram* :

a. *Class Name*

   Nama *class* menggunakan huruf besar diawal kalimatnya dan diletakkan diatas kotak. Bila *class* mempunyai nama yang terdiri dari 2 suku kata atau lebih, maka semua suku kata digabungkan tanpa spasi dengan huruf awal tiap suku kata menggunakan huruf besar.
b. *Attribute*

Attribute adalah *property* dari sebuah *class. Attribute* ini melukiskan batas nilai yang mungkiin ada pada objek yang mungkin ada. Sebuah *class* mungkin mempunyai nol atau lebih *attribute.* Secara konfensi, jika nama *attribute* terdiri atas satu suku kata, maka ditulis dengan huruf kecil. Akan tetapi jika nama *attribute* mengandung lebih darri satu suku kata maka semua suku kata dengan suku kata pertama menggunakan huruf kecil dan awal suku kata berikutnya menggunakan huruf besar.

c. *Operation*

Operation adalah sesuatu yang bias di lakukan oleh sebuah *class* atau *class* yang ain. Seperti halnya *attribute,* nama *operation* juga menggunakan hurup kecil semua terdiri dari satu suku kata. Akan tetapi jika lebih dari satu suku kata, maka semua suku kata digunakan dengan suku kata pertama huruf kecil dan huruf awal tiap suku kata berikutnya dengan hurup besar.

d *Association*

Association adalah konsep dasar hubungan antar *class*. Setiap *class* pada asosiasi memainkan sebuah peran dan *multiplicity* memberikan spesifikasi berapa banyak objek pada suatu *class* berhubungan dengan suatu *classs* pada asosiasi *class.*

2.3 Teori Pendukung

2.3.1 Pengertian Sistem Informasi Administrasi Siswa

Sistem informasi Administrasi Siswa adalah suatu *system* informasi yang terpadu yang merupakan kegiatan yang sangat penting dilembaga pendidikan dan ini merupakan kegiatan yang rutin terjadi disekolah-sekolah. Dimana setiap skala waktu siswa dapat mengurus administrasi terutama surat menyurat dengan ketentuan-ketentuan yang telah disepakati oleh pihak sekolah.

2.3.2 Manajemen Proyek

Manajemen proyek sistem informasi ditekankan pada tiga faktor, yaitu : manusia,masalah dan proses. Dalam pekerjaan sistem informasi faktor manusia sangat berperan penting dalam suksesnya manajemen proyek. Pentingnya faktor

manusia dinyatakan dalam model kematangan kemampuan manajement manusia, yang berfungsi untuk meningkatkan kesiapan organisasi perangkat lunak (sistem informasi) dalam menyelesaikan masalah dengan melakukan kegiatan menerima, memilih, kinerja manajemen, pelatihan, kompensasi, pengembangan karier, organisasi dan rancangan kerja serta pengembangan tim.

Manajemen proyek sistem informasi dalam melakukan pekerjaannya menggunakan sekumpulan organisasi yang merupakan sistem yang terdiri dari beberapa elemen yaitu :

a. orang
b. tujuan
c. posisi
d. pekerjaan
e. teknologi
f. Struktur

Prinsip-prinsip yang ada dalam organisasi, yaitu :

a. tujuan organisasi yang jelas
b. tugas yang dilakukan harus jelas
c. pembagian tugas yang adil
d. penempatan posisi yang tepat
e. adanya koordinasi dan integrasi

Manajemen Proyek Sistem Informasi (MPSI) adalah sebuah cara yang dilakukan untuk mengelola sumber daya (manusia, data, anggaran) untuk mencapai suatu tujuan yang ditentukan. Apa maksud dari mencapai suatu tujuan yang ditentukan? Maksudnya ialah suatu proyek yang dimanajemen sedemikian rupa agar sesuai dengan anggaran, keinginan konsumen, sesuai jadwal, dll. Hal itulah yang menjadi dasar dari manajemen sistem informasi tersebut.

Manajemen Proyek Sistem Informasi (MPSI) bisa juga diartikan sebagai langkah-langkah yang diperlukan dalam sebuah pembuatan proyek sistem

informasi untuk mencapai suatu tujuan yang tadi. Berikut beberapa hal yang dijadikan tujuan dalam hal manajemen sistem informasi:

a. Mutu
b. Biaya
c. Waktu

Jika seorang konsumen memiliki biaya rendah, maka proyek manajer bisa menyesuaikan mutu dengan anggaran yang ada. Jiak konsumen menginginkan pembuatan proyek cepat, maka konsumen harus menyediakan dana lebih untuk mendapatkan tujuan tersebut. Begitu juga dengan tujuan mutu yang dihasilkan.

Dalam hal ini yang mengatur atau mengelola pembuatan sistem informasi dari awal hingga akhir ialah Proyek Manajer. Dimana proyek manajer inilah yang bertanggung jawab dan mengatur segala sesuatu tentang proyek yang dikerjakan.

Seperti: mencari proyek, melakukan estimasi waktu dan biaya, memilih karyawan, dll.

Terdapat 4 dimensi pada pembuatan Sistem Informasi:

a. Manusia, ialah orang yang mengerjakan atau membuat sistem informasi. Mulai dari proyek manajer, hingga *programmer*.
b. Proses, tahap-tahap dimana proyek sistem informasi ini dikerjakan. Pada dimensi inilah dibutuhkan keterampilan seorang proyek manajer untuk mengatur segala sesuatunya agar sesuai dengan tujuan.
c. Produk, ialah hasil dari proyek yang dikerjakan. Dalam hal ini sistem informasi.
d. Teknologi, sesuatu yang terdapat pada produk. Hal ini dapat berkaitan dengan mutu atau kualitas dari sebuah proyek.

Manajemen Proyek adalah kegiatan merencanakan, mengorganisasikan, mengarahkan dan mengendalikan sumber daya organisasi perusahaan untuk mencapai tujuan tertentu dalam waktu tertentu dengan sumber daya tertentu pula. Manajemen proyek sangat cocok untuk suatu lingkungan bisnis yang menuntut

kemampuan akuntansi, fleksibilitas, inovasi, kecepatan, dan perbaikan yang berkelanjutan.

Kegiatan proyek biasanya dilakukan untuk berbagai bidang antara lain sebagai berikut:

Perbaikan fasilitas yang sudah ada. Merupakan kelanjutan dan usaha yang sudah ada sebelumnya. Artinya sudah ada kegiatan sebelumnya, namun perlu dilakukan tambahan atau perbaikan yang diinginkan Pembangunan fasilitas baru. Artinya merupakan kegiatan yang benar-benar baru dan belum pernah ada sebelumnya, sehingga ada penambahan usaha baru.Penelitian dan pengembangan. Merupakan kegiatan penelitian yang dilakukan untuk suatu fenomena yang muncul di masyarakat, lalu dikembangkan sedemikian rupa sesuai dengan tujuan yang diharapkan.

Resiko merupakan bentuk keadaan ketidakpastian tentang suatu keadaan yang akan terjadi nantinya (*future*) dengan keputusan yang diambil berdasarkan berbagai pertimbangan pada saat ini. Manajemen resiko adalah proses pengukuran atau penilaian resiko serta pengembangan strategi pengelolaannya.

Jenis Resiko Teknologi :

1. Komponen *file* tidak lengkap.
2 . Sistem operasi tidak kompatibel, *device* tidak dikenal.
3. Perangkat keras tidak mendukung (mis: resolusi monitor, resolusi printer).
4. Spesifikasi tidak memenuhi.
5. Kualitas *Network* dibawah standar kebutuhan.
6. *Browser, software* tidak memenuhi.

Ada tiga garis besar untuk menciptakan berlangsungnya sebuah proyek, yaitu :

a. Perencanaan

Untuk mencapai tujuan, sebuah proyek perlu suatu perencanaan yang matang. Yaitu dengan meletakkan dasar tujuan dan sasaran dari suatu proyek sekaligus menyiapkan segala program teknis dan administrasi agar

dapat diimplementasikan.Tujuannya agar memenuhi persyaratan spesifikasi yang ditentukan dalam batasan waktu, mutu, biaya dan keselamatan kerja. Perencanaan proyek dilakukan dengan cara studi kelayakan, rekayasa nilai, perencanaan area manajemen proyek (biaya, mutu, waktu, kesehatan dan keselamatan kerja, sumberdaya, lingkungan, resiko dan sistem informasi.

b. Penjadwalan

Merupakan implementasi dari perencanaan yang dapat memberikan informasi tentang jadwal rencana dan kemajuan proyek yang meliputi sumber daya (biaya, tenaga kerja, peralatan, material), durasi dan *progres* waktu untuk menyelesaikan proyek. Penjadwalan proyek mengikuti perkembangan proyek dengan berbagai permasalahannya. Proses *monitoring* dan *updating* selalu dilakukan untuk mendapatkan penjadwalan yang realistis agar sesuai dengan tujuan proyek. Ada beberapa metode untuk mengelola penjadwalan proyek, yaitu Kurva S (hanumm *Curve*), *Barchart*, Penjadwalan *Linear* (diagram Vektor), *Network Planning* dan waktu dan durasi kegiatan. Bila terjadi penyimpangan terhadap rencana semula, maka dilakukan evaluasi dan tindakan koreksi agar proyek tetap berada dijalur yang diinginkan.

c. Pengendalian Proyek

Pengendalian mempengaruhi hasil akhir suatu proyek. Tujuan utama dari utamanya yaitu meminimalisasi segala penyimpangan yang dapat terjadi selama berlangsungnya proyek. Tujuan dari pengendalian proyek yaitu optimasi kinerja biaya, waktu , mutu dan keselamatan kerja harus memiliki kriteria sebagai tolak ukur. Kegiatan yang dilakukan dalam proses pengendalian yaitu berupa pengawasan, pemeriksaan, koreksi yang dilakukan selama proses implementasi.

2.3.3 *Software* Pendukung

2.3.3.1 *Rational Rose*

Rational Rose merupakan sebuah perangkat pemodelan secara *visual* yang memiliki banyak kemampuan (*powerful*) untuk pembentukan sistem berorientasi

obyek yang menggunakan *Unified Modeling Language* (UML). UML merupakan bahasa pemodelan yang dapat digunakan secara luas dalam pemodelan bisnis, pemodelan perangkat lunak dari semua fase pembentukan dan semua tipe sistem, dan pemodelan secara umum dari berbagai pembentukan / konstruksi yang memiliki dua perilaku yaitu baik statis maupun dinamis.

Tutorial ini akan membahas cara pemakaian *Rasional Rose* dengan mengambil sebuah kasus untuk mempermudah pemahaman. Namun demikian tutorial ini bersifat sangat sederhana karena pemakaian perangkat lunak ini sangat ditentukan pada system yang akan dibangun dan variasinya. Tutorial ini dapat dianalogkan dengan kursus privat mengendarai mobil. Mobil merupakan sebuah sarana transportasi yang dapat digunakan untuk berbagai keperluan, dalam kursus *private* hanya diajarkan bagaimana cara mengoperasikan, perpindahan gigi, gas, rem, *light sign*, klakson, dsb. Kemahiran mengendarai ditentukan banyak jam pakai dengan berbagai kasus di jalan dan hal itu tidak diberikan dalam kursus *private* tersebut.

Istilah-istilah yang digunakan dalam UML, bagian-bagian yang digunakan yaitu: *views*, diagram, dan elemen model.

a. *View*. *View* menunjukkan perbedaan dari berbagai aspek-aspek suatu sistem yang dimodelkan. *View* bukan sebuah *graph*, tetapi sebuah abstraksi yang terdiri dari beberapa diagram. Hanya dengan mendefinisikan sejumlah *view*, dimana setiap *view* menunjukkan aspek yang berbeda dan saling terpisah dari sistem, maka gambaran sebuah sistem secara komplit dapat dibentuk. *Rational rose* memiliki empat *view* yaitu: *Use case View, Logical View, Component View,* dan *Deployment View.*

b. Diagram. Diagram merupakan graph yang menjelaskan tentang isi dari sebuah *view*. UML memiliki beberapa tipe diagram yang berbeda yang dapat digunakan untuk mengkombinasi dalam menyusun semua dari sebuah sistem. *Rational Rose* 2000, memiliki delapan diagram yaitu: *Use case diagram, Sequence diagram, Collaboration diagram, Activity Diagram, Class Diagram, Statechart Diagram, Component Diagram* dan *Deployment Diagram.*

c. Elemen Model. Konsep-konsep yang digunakan dalam diagram merupakan elemen-elemen model yang menyatakan konsep-konsep berorientasi obyek secara umum , seperti *class, object,* dan *message*, serta hubungan antar konsep-konsep tersebut termasuk *association, dependency,* dan *generelization.* Sebuah elemen model digunakan dalam beberapa diagram yang berbeda tetapi selalu memiliki simbol dan arti yang sama.

Komponen GUI *Rational Rose* 2000

Komponen utama GUI dari *Rational Rose* diperlihatkan pada gambar dibawah :

a) *Standard toolbar*
b) *Browser*
c) *Diagram window*
d) *Diagram toolbar*
e) *Documentation windows*
f). *Spesification*
g). *Elemen Model (icon)*

2.3.3.2 *Visual Basic* 2008

Microsoft Visual Basic pertama kali diluncurkan pada tahun 1991 dengan nama *Thunder,* yang merupakan *development* pertama yang berbasis *visual* yang dibuat oleh *Microsoft*, untuk menandingi bahasa pemrograman lainnya yang telah ada seperti pemrograman *C, C++, Pascal*, dan bahasa pemrograman lainnya.

Microsoft Visual Basic 2008 merupakan salah satu bahasa pemrograman aplikasi yang sangat dikenal dunia. Aplikasi *Visual Basic* diproduksi pertama kali pada tahun 1991. Pada tahun 1991 *Microsoft* mengeluarkan *Visual Basic* versi 2.0 yang mulai menarik perhatian para pengembang program. Dan ketika versi *Visual Basic 3.0* diluncurkan, versi ini menjadi bahasa pemrograman yang paling pesat perkembangannya di pasaran, sehingga banyak diminati oleh *programmer*. Hal ini membuat jumlah peminatnya menjadi jutaan dan terus bertambah.

Pada tahun 1997, *Microsoft* mengeluarkan *visual basic* versi 5.0 yang memiliki kemampuan untuk menciptakan *Active Control* yang membuat kita mampu menempatkannya di internet dana membuat bahasa *HTML* lebih dinamis dan praktis.

Kemudian pada tahun 1998, *Microsoft* mengeluarkan lagi *Visual Basic* versi 6.0. Dan pada tahun 2008 *Visual Basic* kembali mengeluarkan produk mereka dengan nama *Microsoft Visual Basic 2008*, *Visual Basic* 2008 memiliki tiga dimensi yang hampir sama dengan *Visual Basic 6.0* hanya saja tampilan *Microsoft Visual Basic 2008* lebih menarik dibandingkan dengan *Visual Basic 6.0*. Tiga dimensi yang dimaksud tersebut, yaitu:

a. *Standart Edition*, yang merupakan *Produk Standart* (dasar) yang sudah mencakup berbagai sarana dasar dari *Visual Basic 2008* untuk pengembangan sebuah aplikasi.

b. *Professional Edition*, merupakan versi yang memberikan sarana ekstra yang dibutuhkan oleh *programmer*, misalnya kontrol-kontrol tambahan, dukungan untuk Pemrograman *Internet*, *compiler* untuk membuat *file help*, secara sarana-sarana pengembang *database* yang lebih baik.

c. *Enterprise Edition*, yang memungkinkan *Professional Programmer* untuk membuat aplikasi *client server* yang dapat terhubung ke *internet*. Biasanya versi ini digunakan untuk membuat aplikasi jaringan.

Lingkungan pemrograman *Visual Basic* mengandung semua sarana yang dibutuhkan untuk membangun program-program hebat untuk versi *Windows* dengan cepat dan efisien. *Visual Basic* merupakan bahasa pemrograman yang terstruktur. Struktur Aplikasi *Visual Basic* terdiri dari:

a. *Form* yaitu *Windows* atau jendela dimana *user* akan membuat *User Interface* atau tampilan yang merupakan antar muka program
b. *Control* yaitu tampilan berbasis grafis yang dimasukkan pada *form* untuk membuat interaksi dengan memakai *text, label, option, check, frame,* dan *command.*
c. *Properties* yaitu nilai atau karakter yang dimiliki oleh sebuah objek *Visual Basic*. Contoh: *name, size, color, position*, dan *text*. *Property* dapat diubah saat mendesain program atau *run time* ketika program dijalankan.

d. *Methods* yaitu serangkaian perintah-perintah yang telah tersedia dan dapat diminta untuk melakukan tugas tertentu.
e. *Even procedure* yaitu kode yang berhubungan dengan suatu objek yang dapat diminta mengerjakan tugas khusus. Kode akan dieksekusi ketika ada respon dari pemakai ketika *event* tertentu.
f. *General Procedurs* yaitu kode yang tidak berhuungan dengan suatu objek tetapi pada *general procedures* ini sangat berhubungan dengan aplikasi.
g. *Module* yaitu kumpulan dari prosedur umum, deklarasi variabel dan defenisi konstanta yang digunakan oleh aplikasi.

2.3.3.3 Microsoft Access 2007

Microsoft Access adalah suatu program aplikasi basis data komputer relasional yang digunakan untuk merancang,membuat dan mengolah berbagai jenis data dengan kapasitas yang besar.Aplikasi ini menggunakan mesin basis data Microsoft Jet Database Engine, dan juga menggunakan tampilan grafis yang intuitif sehinnga memudahkan pengguna .Versi terakhir adalah Microsoft Office Access 2007 yang termasuk kedalam Microsoft Office System 2007.*Database relationship* adalah relasi atau hubungan antara beberapa tabel dalam database yang kita miliki. Relasi antar tabel dihubungkan oleh *primary key* dan *foreign key*. Untuk membuat relationship maka masing-masing tabel harus memiliki *primary key* dan *foreign key* untuk dapat menghubungkan antara tabel induk dengan tabel anak maka penulis menggunakan *microsoft access* 2007.