BAB II
LANDASAN TEORI

2.1 Pengertian Sistem dan Informasi

2.1.1 Pengertian Sistem

Sistem adalah sekelompok elemen yang terintegrasi dengan maksud yang sama untuk mencapai suatu tujuan organisasi atau perusahaan yang terdiri dari sejumlah sumber data (manusia, material, mesin, uang, informasi) yang ditentukan oleh pemilik atau manajemen.

2.1.2 Pengertian Informasi

Menurut Chr. Jimmy L.Gaol (2008), Informasi adalah segala sesuatu keterangan yang bermanfaaat untuk para pengambilan keputusan /manajer dalam rangka mencapai tujuan organisasi yang sudah ditetapkan sebelumnya. Untuk dapat berguna, maka informasi harus didukung oleh tiga pilar, yaitu sebagai berikut :

a. Relevan (*Relevancy*), dalam hal ini informasi yang diterima harus memberikan manfaat bagi pemakainya. Kadar *relevancy* informasi antara orang satu dengan yang lainnya berbeda-beda tergantung kepada kebutuhan masing-masing pengguna informasi tersebut. *How is the message used for problem solving (decision masking)?*
b. Akurat (*Accurate*), yaitu berarti informasi harus bebas dari kesalahan-kesalahan. Selain itu informasi yang didapatkan tidak boleh bias atau menyesatkan bagi penggunanya, serta harus dapat mencerminkan dengan jelas maksud dari informasi tersebut. Ketidak akuratan data terjadi karena sumber dari informasi tersebut mengalami gangguan dalam penyampaiannya baik hal itu dilakukan secara sengaja maupun tidak, sehingga menyebabkan data asli tersebut berubah atau rusak.
c. Tepat waktu (*TimeLines*), Informasi yang dibutuhkan oleh si pemakai dalam hal penyampaiannya tidak boleh terlambat (usang). Karena informasi yang usang maka informasi tersebut tidak mempunyai nilai yang baik dan

kualitasnya pun menjadi buruk sehingga tidak berguna lagi. Jika informasi tersebut digunakan sebagai dasar pengambilan keputusan, maka akan berakibat fatal sehingga salah dalam pengambilan keputusan tersebut. Kondisi tersebut mengakibatkan mahalnya nilai suatu informasi, sehingga kecepatan untuk mendapatkan, mengolah serta mengirimnya memerlukan teknologi terbaru.

d. Ekonomis (*Economy*), *What level of resources is needed to move information through the problem-solving cycle?* Kualitas dari Informasi yang digunakan dalam pengambilan keputusan juga bergantung pada nilai ekonomi yang terdapat didalamnya.

e. Efisien (*Efficiency*) *What level of resources is required for each unit of information output ?*

f. Dapat dipercaya *(Reliability)*, Informasi yang didapatkan oleh pemakai harus dapat dipercaya, hal ini menentukan terhadap kualitas informasi serta dalam hal pengambilan keputusan setiap tingkatan manajemen.

2.2 Pengertian Sistem Informasi

Menurut Tata Sutabri, S.Kom,MM (2004: 36), Sistem informasi dapat diartikan sebagai suatu sistem didalam suatu organisasi yang mempertahankan kebutuhan pengolahan transaksi harian yang mendukung fungsi operasi organisasi yang bersifat manajerial dengan kegiatan strategi dari suatu organisasi untuk dapat menyediakan kepada pihak luar tertentu dengan laporan-laporan yang diperlukan.

2.3 Teori Pendukung

2.3.1 Pengertian Akademik

Menurut Kamus Besar Bahasa Indonesia akademik adalah lembaga pendidikan tinggi kurang lebih 3 tahun berhubungan dengan akademik, bentuk ilmiah, bersifat ilmu pengetahuan.

2.3.2 Pengertian Sistem Informasi Akademik

Sistem Informasi akademik merupakan tiang utama dalam mengatur segala hal yang berkaitan dengan penyelenggaraan kegiatan, didalam sistem inilah

komponen – komponen yang ada dapat saling berinteraksi. Sebuah system informasi akademik yang baik tentunya mampu menjalankan semua hal yang berkaitan dengan penyelenggaraan maupun hal – hal spesifik lainnya, semua komponen dipermudah dengan adanya system ini, tidak perlu terjadi kesalahpahaman jika aturan – aturannya sudah masuk kedalam sistem.

Sistem Informasi Akademik (SIAKAD) adalah suatu sistem yang dirancang untuk keperluan pengeloaan datadata akademik dengan penerapan teknologi komputer baik *hardware* maupun *software* yang bertujuan memberikan informasi terhadap para siswa, orang tua siswa dan masyarakat tentang sekolah, fasilitas sekolah, data siswa, absensi siswa, data prestasi siswa, data nilai siswa, dan pembayaran sekolah, jadwal guru piket, dan saran.

2.4 Manajemen 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. Manajemen proyek mempergunakan personel perusahaan untuk ditempatkan pada tugas tertentu dalam proyek. Beberapa hal yang terdapat dalam manajemen proyek adalah sebagai berikut:

2.4.1 Manajemen Waktu

Manajemen Waktu merupakan perencanaan, pengorganisasian, penggerakan, dan pengawasan produktivitas waktu. Waktu menjadi salah satu sumber daya untuk kerja. Sumber daya yang mesti dikelola secara efektif dan efisien.

2.4.2 Manajemen Biaya

Manajemen Biaya adalah suatu bentuk akuntansi manajemen yang memungkinkan sebuah bisnis untuk memprediksi pengeluaran yang akan datang untuk membantu mengurangi kemungkinan akan melebihi anggaran.

2.4.3 Manajemen Sumber Daya

Menurut Hasibuan (2000: 10) Manajemen sumber daya manusia adalah ilmu dan seni mengatur hubungan dan peranan tenaga kerja agar efektif dan efisien membantu terwujudnya tujuan perusahaan, karyawan dan masyarakat.

2.4.4 Manajemen Pemasaran

Pengertian manajemen pemasaran menurut Sofyan Assauri(2004) Manajemen Pemasaran merupakan kegiatan menganalisis, merencanakan, mengkoordinasikan dan mengendalikan semua kegiatan yang terkait dengan perancangan dan peluncuran produk, pengkomunikasian, promosi dan pendistribusian produk tersebut, menetapkan harga dan mentransaksikannya, dengan tujuan agar dapat memuaskan konsumennya dan sekaligus dapat mencapai tujuan organisasi perusahaan jangka panjang.

2.4.5 Manajemen Resiko

Pengertian manajemen resiko menurut Tampubolon (2004), Manajemen risiko juga dapat diartikan sebagai kegiatan atau proses yang terarah dan bersifat proaktif, yang ditujukan untuk mengakomodasi kemungkinan gagal pada salah satu, atau sebagian dari sebuah transaksi atau instrumen.

2.5 Analisa Dan Perancangan Sistem Berorientasi Objek *dengan Unified Modeling Language*

2.5.1 Konsep Dasar Berorientasi Objek

Obyek adalah "benda" secara fisik atau konseptual, yang dapat kita temui di sekeliling kita. Obyek adalah riil. Contoh obyek adalah orang, *hardware*, *software*, dokumen dan lain-lain.

Setiap obyek mempunyai dua ciri, yaitu atribut (*property* atau data) yang menjadi ciri khas dari suatu obyek (*what they have*) dan *method* (*behavior/function*), yaitu apa yang dapat dilakukan oleh obyek (*what they do*).

Berorientasi Obyek (*object oriented*) berarti permasalahan didefinisikan melalui istilah dari obyek yang mengkapsulasi data (attribut) dan perilaku (*behavior*), yaitu melalui paradigma/pendekatan obyek.

Selain *object*, ada beberapa istilah yang akan membantu untuk memahami pengertian kita dalam skripsi ini:

a. *Class*, yaitu kumpulan obyek yang sejenis. Secara lebih lugas obyek adalah *instant* dari sebuah *class*, atau dengan pengertian lain dengan *class* kita menggambarkan *property* dan *behavior* dari tipe obyek.
b. *Inheritance*, adalah penurunan atribut atau method dari suatu obyek *class* ke obyek *class* lainnya.
c. *Polymorphisme*, berasal dari bahasa Yunani yang berarti banyak bentuk.

Dalam konsep ini memungkinkan digunakannya suatu *interface* yang sama untuk memerintah suatu obyek untuk melakukan suatu aksi atau tindakan yang mungkin secara prinsip sama tetapi secara proses berbeda. Secara sederhana bisa juga disebut: satu *interface*, banyak aksi.

Metodologi adalah cara sistematis untuk mengerjakan pekerjaan analisis dan desain. Metodologi berorientasi obyek adalah metode penyelesaian masalah dengan menggunakan pendekatan berorientasi obyek.

Metodologi berorientasi obyek pertama kali muncul pada pertengahan tahun 1970 dan terus berkelanjutan dikembangkan sampai saat ini. Pada tahun 1994 ada 72 lebih metode *object oriented*. Dengan berkembang pesatnya metode ini maka masyarakat *object oriented* menyadari perlunya standarisasi.

2.5.2 UML (*Unified Modelling Language*)

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 obyek (*object-oriented analysis and design methode*) yang berkembang pada era 80-an sampai 90-an. Pada masa itu, banyak metode

berorientasi obyek 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) bergabung 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 Methode* 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 untuk aplikasi berorientasi objek.

2.5.3 Analisa dan Perancangan Berorientasi Objek

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).

Tujuan perancangan sistem itu untuk memahami kebutuhan kepada pemakai sistem (*user*) dan memberikan gambaran yang jelas serta rancang bangun yang lengkap.

Perancangan berorientasi obyek bertujuan untuk :

a. Sistematika proses pendisaian
b. Menghasilkan pendisaian model program

c. Memberikan gambaran pemecahan masukan dengan efektif

2.5.4 Analisa Berorientasi Objek (*Object Oriented Analysis*)

Object oriented analysis adalah metode analisis yang memeriksa *requirements* (syarat atau keperluan yang harus dipenuhi suatu sistem). (Suhendara dan Hariman, 2002: 11)

Dalam tahap ini kegiatan-kegiatan yang dilakukan dalam menganalisa sistem sebagai berikut :

a. Menganalisa sistem yang ada dan mempelajari apa yang dikerjakan oleh sistem yang ada.
b. Menspesifikasikan sistem yaitu spesifikasi masukan yang digunakan *database* yang ada, proses yang dilakukan dan keluaran yang dihasilkan.

Tujuan dari analisa berorientasi obyek yaitu untuk menentukan kebutuhan pemakai secara akurat.

Pendekatan-pendekatan yang dipakai dalam analisa berorientasi obyek antara lain :

a. Pendekatan *top down*, yaitu memecahkan masalah ke dalam bagian-bagian terkecil atau per level sehingga mudah untuk diselesaikan.
b. Pendekatan modul, yaitu membagi sistem ke dalam modul-modul yang dapat beroperasi tanpa ketergantungan.
c. Penggunaan alat-alat bantu dalam bentuk grafik dan teks sehingga mudah untuk dimengerti serta dikoreksi apabila terjadi perubahan.

Pendekatan dalam analisa berorientasi obyek dilengkapi dengan alat-alat dan teknik-teknik yang dibutuhkan dalam pengembangan sistem, sehingga hasil akhir dari sistem yang dikembangkan akan didapatkan sistem yang terdefinisi dengan baik dan jelas.

2.5.4.1 *Activity* Diagram

Diagram aktivitas menggambarkan proses bisnis dan urutan aktivitas-aktivitas yang mendukung penggambaran tindakan sistem baik yang bersifat kondisional maupun paralel. Tindakan kondisional dilukiskan dengan cabang (*branch*) dan penyatuan (*merge*).

Sebuah *branch* memiliki sebuah *transition* masuk atau yang disebut dengan *incoming transition* dan beberapa *transition* keluar atau yang disebut dengan *outgoing transition* dari *branch* yang berupa keputusan-keputusan. Hanya satu dari *outgoing transition* yang dapat diambil, maka keputusan-keputusan tersebut harus bersifat *mutually exclusive*. [*else*] digunakan sebagai keterangan singkat yang menunjukkan bahwa transition "*else*" tersebut harus digunakan jika semua keputusan yang ada pada *branch* salah.

Sebuah *merge* memiliki banyak input *transition* dan sebuah *output*. *Merge* menandakan akhir dari suatu kondisi yang diawali dengan sebuah *branch*. Selain *branch* dan *merge*, di dalam diagram aktivitas terdapat pula *fork* dan *join*. *Fork* memiliki satu *incoming transition* dan beberapa *outgoing transition*. Sedangkan pada *join*, *outgoing transition* diambil atau digunakan hanya ketika semua *state* pada *incoming transition* telah menyelesaikan aktivitasnya.

Activity diagram adalah teknik 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 pararel sedangkan *flowchart* tidak bisa. (Munawar 109)

Simbol-simbol yang sering digunakan pada saat pembuatan *activity* diagram adalah sebagai berikut:

a. *Start Point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas. (Munawar 109)

Simbol *start point*

b. *End Point*, akhir aktifitas. (Munawar 109)

Simbol *end point*

c. *Activity*, menggambarkan suatu proses/kegiatan bisnis. (Munawar 109)

Simbol *activity*

d. *Fork* (percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara pararel atau untuk menggabungkan dua kegiatan pararel menjadi satu. (Munawar 110)

Simbol *fork*

e. *Decision Points*, menggambarkan pilihan untuk pengambilan keputusan, *true* atau *false*. (Munawar 110)

Simbol *decision points*

f. *Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa

Simbol *swimlane*

2.5.4.2 Analisa Dokumen Keluaran

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

2.5.4.3 Analisa Dokumen Masukan

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

2.5.4.4 *Package* Diagram

Package (paket) adalah mekanisme pengelompokan yang digunakan untuk menandakan pengelompokan elemen-elemen model. Sebuah *package* dapat mengandung beberapa paket lain di dalamnya. *Package* digunakan untuk memudahkan pengorganisasian elemen-elemen model.

2.5.4.5 *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 :

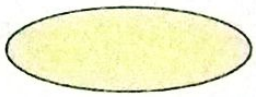
a. Actor

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 :

Simbol *Actor*

b. *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*).



Simbol *Use Case*

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*.

c. *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.

Simbol *Association*

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

1) *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.

Simbol *Association* Antar *Actor* dan *Use Case*

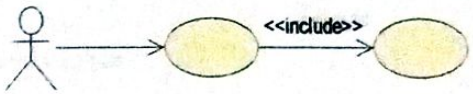
Association antara *actor* dan *use case* sebaiknya menggunakan garis tanpa panah. *Association* antar *actor* dan *use case* yang menggunakan panah terbuka untuk mengindikasikan bila *actor* berinteraksi secara pasif dengan sistem.

2) *Association* antara *use case*

(a) *<<include>>*

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.*



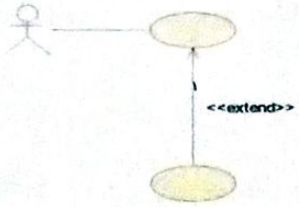
Gambar 2.1

Contoh *Include*

(b) *<<extend>>*

Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*nya sendiri. Sementara hubungan generalisasi antar *use*

case menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.



Gambar 2.2

Contoh *Extend*

2.5.5 Perancangan Berorientasi Objek (*Object Oriented Design*)

Object oriented design adalah metode untuk mengarahkan arsitektur *software* yang didasarkan pada manipulasi obyek-obyek sistem atau subsistem (Suhendar dan hariman 200: 11). Perancangan berorientasi obyek merupakan proses spesifikasi yang terperinci atau pendefinisian dari kebutuhan-kebutuhan fungsional dan persiapan untuk rancang bangun implementasi yang menggambarkan bagaimana suatu sistem dibentuk. Untuk mengembangkan suatu sistem baru digunakan dengan menguraikan hubungan proses-proses dalam bentuk diagram-diagram.

Perancangan berorientasi obyek bertujuan untuk :

a. Sistematika proses pendesainan
b. Menghasilkan pendesainan model program
c. Memberikan gambaran pemecahan masukan dengan efektif

Tahap-tahap yang dilakukan dalam perancangan berorientasi obyek adalah sebagai berikut :

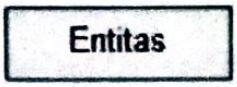
2.5.5.1 Perancangan Basis Data

Merupakan tahap merancang basis data yang akan diterapkan oleh sistem. Berbeda dengan langkah-langkah yang dilakukan dalam perancangan sistem terstruktur, secara garis besar tahap dalam merancang basis data pada perancangan berorientasi obyek sebagai berikut :

Entity Relationship Diagram (ERD), digunakan untuk menggambarkan dan menjelaskan tentang hubungan antara penyimpanan data (data *store*) yang ada di dalam diagram aliran data. Komponen-komponen yang digunakan antara lain sebagai berikut :

a. *Entity Set*

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.



Simbol *entity class*

b. *Relationship Set*

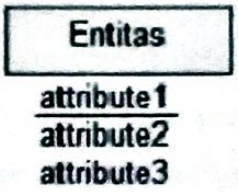
Pada *Entity Relationship* Diagram (ERD) setiap *relationship set* digambarkan dengan sebuah bentuk belah ketupat, dengan garis yang menghubungkan satu *entity* dengan *entity* lain yang terkait. *Relationship set* menunjukkan hubungan alamiah yang terjadi pada *entity*. *Relationship set* adalah kumpulan *relationship* yang sejenis. Pada umumnya *relationship set* diberi nama dengan kata kerja.

Gambar 2.3

Simbol *relationship*

c. *Attribute*

Setiap entitas pasti mempunyai elemen yang disebut *atribute* 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.



Simbol Attribut

Setiap entitas harus memiliki atribut yang unik untuk pengidentifikasian, atribut yang dimaksud disebut dengan *key*. Jenis-jenis *key* adalah:

1) *Primary Key*, yaitu *key* yang paling umum digunakan untuk mengidentifikasi secara unik setiap instansi dari entitas. (Whitten 298)
2) *Secondary Key*, yaitu suatu *key* yang tidak terpilih untuk dijadikan *primary key* juga disebut sebagai *alternate key*. (Whitten 298)
3) *Composite key*, suatu kelompok atribut yang dapat dengan unik mengidentifikasi suatu instansi dari suatu entitas. (Whitten 297)
4) *Foreign key*, suatu *primary key* dari suatu entitas yang digunakan di entitas yang lain untuk mengidentifikasi instansi dari suatu hubungan (*relationship*). (Whitten 301)

d. Cardinality

Cardinality adalah tingkat hubungan antara entitas dan dilihat dari segi kejadian atau banyak tidaknya hubungan yang terjadi antara *entity* pada ERD. Ada tiga kemungkinan tingkat hubungan yang ada, yaitu :

1) *One To One* (1:1)

   Terjadi bila suatu entitas hanya memiliki sebuah hubungan dengan entitas lainnya dan hubungan dinyatakan satu pada satu kejadian.
2) *One To Many* atau *Many To One* (1:M, M;1)

   Terjadi apabila sebuah entitas memiliki banyak hubungan dengan entitas lain atau sebaliknya.
3) *Many* To *Many* (M:N)

   Terjadi apabila dua buah entitas memiliki banyak hubungan.

2.5.5.2 *Logical Record Structure* (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 *type 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).

2.5.5.3 Tabel

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 ER 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.

2.5.5.4 Normalisasi

Normalisasi adalah kegiatan mengelompokkan atribut-atribut sehingga terbentuk relasi berorientasi obyek dengan baik. Normalisasi juga merupakan

proses untuk mengorganisasi *file* dengan menghilangkan *group* elemen atau proses menyederhanakan *relationship* antar elemen data di dalam *tuple* (*record*).

Normalisasi banyak dilakukan dalam mengubah bentuk *database* dari suatu struktur jaringan menjadi struktur hubungan. Konsep dan teknik normalisasi pertama kali diperkenalkan oleh Dr. E.F.Codd dalam bentuk struktur hubungan. Istilah data hubungan menunjukkan suatu obyek data yang mempunyai hubungan dengan elemen-elemen data lainnya, baik dalam suatu *file* atau dalam *file* yang lain.

Tahapan-tahapan dalam normalisasi :

a. Normalisasi bentuk pertama (*First* Normal *Form* / 1NF)
   Yaitu menghilangkan beberapa group elemen berulang (*repeating group*) agar menjadi satu harga tunggal yang berinteraksi di antara setiap baris dan kolom pada suatu tabel dikatakan sudah berada pada 1 NF jika dan hanya jika semua nilai atributnya adalah *atomic* (tunggal).
b. Normalisasi bentuk kedua (*Second* Normal *Form* / 2NF)
   Yaitu menghilangkan beberapa bagian ketergantungan fungsional (*functional dependency*) atau dengan kata lain apabila sudah berada pada 1 NF dan setiap atribut yang bukan *key*, *full functional dependency* terhadap *primary key*.
c. Normalisasi bentuk ketiga (*Third* Normal *Form* / 3NF)
   Yaitu menghilangkan beberapa bentuk ketergantungan *transitive* (*transitive dependency*) atau dengan kata lain apabila sudah berada pada 2NF dan setiap atribut yang bukan *key* tidak tergantung pada atribut lain (tidak transitif) kecuali terhadap *primary key (non transitively dependent* terhadap *primary key*). Pada umumnya dalam tahap ini sudah memenuhi syarat untuk sebagian besar aplikasi *database*.
d. *Boyce-Codd* Normal *Form* (BCNF)
   Yaitu menghilangkan terdapatnya anomaly pada relasi yang disebabkan oleh *overlapping candidate key* atau apabila setiap determinan adalah merupakan *candidate key*.
e. Normalisasi bentuk keempat (*Fourth* Normal *Form* / 4NF)

Yaitu menghilangkan beberapa ketergantungan pada banyak harga (*multivalue dependency*).

f. Normalisasi bentuk kelima (*Fifth* Normal *Form* / 5NF)

Yaitu *join dependency* anomaly yang terjadi akibat dekomposisi relasi tidak dapat dipakai kembali untuk membentuk kembali relasi semula.

2.5.5.5 Spesifikasi Basis Data

Basis data merupakan kumpulan dari data yang saling berhubungan satu dengan yang lain dan tersimpan di luar komputer serta digunakan perangkat lunak (*software*) tertentu untuk memanipulasinya.

Sedangkan sistem basis data adalah suatu sistem penyusunan dan pengelolaan *record-record* dengan menggunakan komputer dengan tujuan untuk menyimpan atau merekam serta melihat data operasional lengkap pada sebuah organisasi atau perusahaan, sehingga mampu menyediakan informasi optimal yang diperlukan untuk kepentingan proses pengambilan keputusan.

2.5.5.6 Rancangan Dokumen Keluaran

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

2.5.5.7 Rancangan Dokumen Masukan

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

2.5.5.8 Rancangan Layar Program

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

2.5.5.9 Sequence Diagram

Diagram yang menggambarkan bagaimana obyek berinteraksi dengan obyek lainnya melalui pesan (*message*) yang disampaikan, disusun dalam urutan kejadian atau waktu dan secara khusus berasosiasi dengan *use case*.

Setelah kita menentukan tanggung jawab dan perilaku dari obyek, kita dapat menciptakan suatu model yang terperinci dari bagaimana obyek tersebut akan saling berhubungan satu sama lain untuk menyediakan kemampuan/fungsi yang ditetapkan pada setiap *use case* yang telah didesain sebelumnya. UML menyediakan dua jenis diagram untuk melukiskan interaksi tersebut dengan nyata. *Sequence* diagram dan *collaboration* diagram.

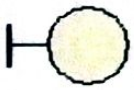
Sequence diagram menunjukkan secara *detail* bagaimana obyek saling berhubungan satu sama lain dari waktu ke waktu, dan *collaboration diagram* menunjukkan bagaimana obyek bekerja sama dalam bentuk urutan pesan untuk memenuhi fungsionalitas dari suatu *use case*.

Definisi dari *sequence* diagram adalah suatu diagram UML yang memodelkan logika dari suatu *use case* dengan menggambarkan interaksi berupa pengiriman pesan(*message*) antar obyek dalam urutan waktu. (Whitten: 702). Beberapa simbol yang umum digunakan pada *sequence* diagram, yaitu:

a. *Entity Object*, suatu obyek yang berisi informasi kegiatan yang terkait yang tetap dan disimpan ke dalam suatu *database*. (Whitten 686)

Simbol *entity object*

b. *Interface/Boundary Object*, sebuah obyek yang menjadi penghubung antara *user* dengan sistem. Contohnya *window*, *dialogue box* atau *screen*(tampilan layar). (Whitten 686)

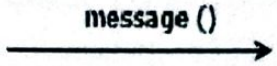


Simbol *boundary object*

c. *Control Object*, suatu obyek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas. Contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai obyek. *Control object* mengkoordinir pesan(*message*) antara *boundary* dengan entitas. (Whitten 686)

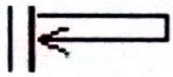
Simbol *control object*

d. *Simple Message*, simbol pengiriman pesan dari sebuah obyek ke obyek lain. (Whitten 704)



Simbol *message*

e. *Recursive*, sebuah obyek yang mempunyai sebuah *operation* kepada dirinya sendiri. (Munawar 89)



Simbol *recursive*

f. *Activation*, *Activation* mewakili sebuah eksekusi operasi dari obyek, panjang kotak ini berbanding lurus dengan durasi aktivasi sebuah operasi. (Munawar 87-89)

Simbol *activation*

g. *Lifeline*, garis titik-titik yang terhubung dengan obyek, sepanjang *lifeline* terdapat *activation*. (Munawar 87-89)

Simbol *lifeline*

2.5.5.10 *Class* Diagram

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

Contoh *Class* Diagram

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 biasa di lakukan oleh sebuah c*lass* atau *class* yang lain. Seperti halnya *attribute,* nama *operation* juga menggunakan huruf 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 huruf 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*.