BAB II
LANDASAN TEORI

Pada bab ini berisi penjelasan tentang teori yang menjadi landasan dalam penyusunan tugas akhir ini. Dan secara garis besar akan dijelaskan mengenai pengertian dan konsep-konsep dasar yang akan digunakan dalam perancangan sistem yang akan dibuat dalam tugas akhir ini.

1. Konsep Sistem Informasi

Dalam sistem informasi adanya inti dan tujuan yang menghasilkan informasi itu sendiri. Walaupun sesederhana mungkin sistem informasi yang dikembangkan, jika bisa menghasilkan informasi yang diharapkan tentunya pengembangan sistem informasi tersebut akan dikatakan berhasil. Namun disisi lain, secanggih apapun sistem informasi yang dikembangkan bila tidak menghasilkan informasi yang diharapkan, maka pengembangannya dikatakan gagal. Adapun penjelasan mengenai konsep dasar sistem informasi sebagai berikut.

a. Konsep Dasar Sistem dan Informasi

Pengertian sistem menurut Hart 2005 yaitu sistem mengandung dua pengertian utama yaitu: (a) pengertian sistem yang menekankan pada komponen atau elemennya yaitu sistem merupakan komponen-komponen atau subsistem yang saling berinteraksi satu sama lain,dimana masing-masing bagian tersebut dapat bekerja secara sendiri-sendiri (independent) atau bersama-sama serta saling berhubungan membentuk satu kesatuan sehingga tujuan atau sasaran sistem tersebut dapat tercapai secara keseluruhan. (b) definisi yang menekankan pada prosedurnya yaitu merupakan suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran tertentu[Fairuz 2010].

Menurut Kerz 2008 Sistem yaitu gabungan dari sekelompok komponen baik itu manusia ataupun bukan manusia(non-human) yang saling mendukung satu sama lain serta diatur menjadi sebuah kesatuan yang utuh untuk mencapai suatu tujuan, sasaran atau hasil akhir[Fairuz 2010].

Adapun karakteristik atau sifat-sifat tertentu dari sistem,yaitu:

1) **Komponen,** dapat berupa Elemen-elemen yang lebih kecil yang disebut sub sistem, misalkan sistem komputer terdiri dari sub sistem perangkat keras, perangkat lunak dan manusia. Elemen-elemen yang lebih besar yang disebut supra sistem
2) **Batas sistem,** merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainnya atau dengan lingkungan luarnya. Batas sistem ini memungkinkan suatu sistem dipandang sebagai suatu kesatuan. Batas suatu sistem menunjukkan ruang lingkup dari sistem tersebut.
3) **Lingkungan luar sistem,** adalah apapun di luar batas dari sistem yang mempengaruhi operasi sistem. Lingkungan luar sistem dapat bersifat menguntungkan dan dapat juga bersifat merugikan sistem tersebut.
4) **Penghubung,** merupakan media perantara antar subsistem. Melalui penghubung ini memungkinkan sumber-sumber daya mengalir dari satu subsistem ke subsistem lainnya.
5) **Masukkan,** merupakan energi yang dimasukkan ke dalam sistem
6) **Keluaran,** adalah hasil dari energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna dan sisa pembuangan.
7) **Pengolah**, Suatu sistem dapat mempunyai suatu bagian pengolah atau sistem itu sendiri sebagai pengolahnya.
8) **Sasaran atau tujuan,** Suatu sistem pasti mempunyai tujuan atau sasaran. Kalau suatu sistem tidak mempunyai sasaran, maka operasi sistem tidak akan ada gunanya. Sasaran dari sistem sangat

menentukan sekali masukan yang dibutuhkan sistem dan keluaran yang akan dihasilkan sistem. Suatu sistem dikatakan berhasil bila mengenai sasaran atau tujuannya.

b. Konsep Sistem Informasi

Menurut [Al-Bahra Bin Ladjamudin 2005] Sumber informasi adalah data. Data merupakan kenyataan yang menggambarkan suatu kejadian-kejadian dan suatu kenyataan. Kejadian adalah suatu yang terjadi pada saat tertentu sedangkan informasi diperoleh setelah data-data mentah diproses atau diolah.

Kegunaan informasi adalah untuk mengurangi ketidakpastian dalam pengambilan keputusan tentang suatu keadaan. Sistem informasi umumnya digunakan untuk beberapa kegunaan dan tidak hanya oleh satu orang pihak didalam organisasi. Nilai sebuah informasi ditentukan dari dua hal yaitu manfaat dan biaya untuk mendapatkan informasi tersebut.

Menurut [Aji supriyanto 2005] “sistem informasi adalah suatu sistem di dalam suatu organisasi, yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan”. Sedangkan pendapat [Al-bahra bin ladjamudin 2005] mengatakan “sistem informasi adalah sekumpulan prosedur organisasi yang pada saat dilaksanakan akan memberikan informasi bagi pengambil keputusan dan atau mengendalikan organisasi”.

Sistem informasi banyak digunakan berbagai pihak antara lain organisasi menggunakan sistem informasi untuk mengolah transaksi-transaksi, mengurangi biaya dan menghasilkan pendapatan sebagai salah satu produk atau pelayanan mereka. Bank menggunakan sistem informasi untuk mengolah cek-cek nasabah dan membuat berbagai laporan rekening koran dan transaksi yang terjadi. Perusahaan menggunakan

sistem infomasi untuk mempertahankan persediaan pada tingkat paling rendah agar konsisten dengan jenis barang yang tersedia.

Kelima komponen tersebut dapat diklasifikasikan sebagai berikut :

1) *Hardware* dan *software* yang berfungsi sebagai mesin.
2) *People* dan *procedures* yang merupakan manusia dan tatacara menggunakan mesin.
3) Data merupakan jembatan penghubung antara manusia dan mesin agar terjadi suatu proses pengolahan data.

Sistem infomasi terdiri dari komponen-komponen yang disebut sebagai blok bangunan (*building block*). Blok bangunan tersebut terdiri dari blok masukan (*input blok*), blok model (*model block*), blok keluaran (*output block*), blok teknologi (*technology block*), blok basis data (*database block*), dan blok kendali (*controls block*).

Menurut [Al-Bahra Bin Ladjamudin 2005] Informasi harus memenuhi kriteria sebagai berikut :

1) Informasi harus akurat, sehingga mendukung pihak manajemen dalam mengambil keputusan.
2) Informasi harus relevan, benar benar terasa manfaatnya bagi yang membutuhkan.
3) Informasi harus tepat waktu, sehingga tidak ada keterlambatan pada saat dibutuhkan.

2. Analisa dan Perancangan Sistem Berorientasi Obyek Dengan UML

Analisa dan perancangan berorientasi obyek berarti merumuskan dan menyelesaikan masalah serta mengasilkan suatu *hipotesa* atau *diagnosa* (solusi), memodelkannya dengan pendekatan atau paradigma obyek (obyek adalah suatu riil yang mempunyai atribut atau data dan perilaku).

a. UML (*Unified Modeling Language*)

Menurut ***Sun Microsystem Inc.***, dalam buku tutorial panduan siswa dinyatakan " *The unifiedModelling (UML) is a graphical language for*

visualizing, specipying, constructing and documenting the artifacts of a software-intensive system". (UML adalah bahasa nyata (grafis) yang menggambarkan, menetapkan, membangun, dam mendokumentasikan sesuatu (benda) pada sebuah sistem perangkat secara intensif).

UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunaka model, diharapkan pengembangan *software* dapat memenuhi semua kebutuhan pengguna dengan lengkap dan tepat, termasuk faktor-faktor seperti lingkup (*scalability*), kemampuan (*robustness*), keamanan (*security*), dan sebagainya. Pemodelan adalah proses merancang *software* sebelum melakukan pengkodean (*coding*). Dengan menggunakan UML kita dapat memuat model untuk semua aplikasi *software*, dimana aplikasi tersebut dapat berjalan pada *hardware*, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemprograman apapun. Tetapi karena UML juga menggunakan kelas dan operasi dalam konsep dasarnya, maka ia lebih cocok ntuk penulisan *software* dalam bahasa-bahasa pemprograman berorientasi obyek seperti C++, Java, *smalltalk.*

Seperti bahasa lainnya, UML mendefinisiskan notasi dan sintaksis/ semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram *software*. Setiap bentuk memiliki makna tertentu, dan sintaksis UML mendefinisikan bagaimana bentuk-bentuk tersebut dapat didefinisikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumya *: Object-Oriented Design* (OOD) dari Grady Booch, *Object Modeling Technique* (OMT) dari jim Rumbaugh, dan Ivar Jacobson *Object-Oriented Software Engineering* (OOSE) dari Jacobson.

Beberapa metode analisa berorientasi obyek yang popular adalah metodologi Booch (*Object Oriented-Design*), metode coad dan yourdon, metodologi Jacobson (*Object Oriented software Engineering*-OOSE), metodologi Rumbaugh (*Object Modelling Technique*-OMT), metode wirfs-brock dan metode *Rational Unified Proccess*. Masa itu terkenal

dengan masa perang metodologi (*method war*) dalam pendesainan berorientasi obyek.

Pengembangan UML dimulai pada oktober 1994 oleh Booch dan Rumbaugh. Jacobson bergabung pada musim gugur 1995. Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh *object management group (OMG – http : / www.omg.org).* Tahun 1997 UML versi 1.1 muncul, dan muncul versi 1.5 yang dirilis bulan maret 2003, dan yang terbaru adalah versi 2.0. UML telah menjadi standar bahasa pemodelan untuk aplikasi berorientasi obyek..

Tujuan utama dibentuknya UML diantaranya adalah untuk :

1) Memberikan model yang siap pakai, bahasa pemodelan visual yang ekpresif untuk mengembangkan dan saling menukar model dengan mudah dan dimengerti secara umum.
2) Menyatukan praktek-praktek terbaik yang terdapat dalam pemodelan.
3) Memberikan bahasa pemodelan yang bebas dari berbagai bahasa pemprograman dan proses rekayasa.

Untuk membuat suatu model, UML mendefinisikan diagram-diagram sebagai berikut :

1) Use Case Diagram
2) Activity Diagram
3) Sequence Diagram
4) Communication Diagram
5) Class Diagram
6) Component Diagram
7) Deployment Diagram
8) State Machine Diagram
9) Composite Structure Diagram
10) Interaction Overview Diagram
11) Object Diagram
12) Package Diagram

13) Timing Diagram

Dalam menganalisa dan merancang sistem yang ditulis dalam tugas akhir ini, penulis tidak menggunakan semua alir diagram, hanya beberapa saja yang dibutuhkan seperti *use case diagram, activity diagram, class diagram* dan beberapa lagi yang digunakan pada masing-masing proses analisa dan perancangan.

b. Analisa Sistem Berorientasi Objek

Menurut [Aji supriyanto 2005] "Analisa sistem berorientasi obyek adalah tahap menentukan kebutuhan perangkat lunak, yang mendaftarkan apa pun yang harus dipenuhi oleh sistem *software*, bukan mengenai bagaimana sistem *software* melakukannya". Hasil dari tahap analisa adalah dokumen *software requirement specification* (SRS).

Yang dilakukan dalam analisa berorientasi obyek adalah mempelajari domain permasalahan, kemudian menghasilkan spesifikasi dari tingkah laku eksternal yang diamati akan mempengaruhi dan mendukung domain perusahaan. Analisa berorientasi obyek yang baik adalah yang merupakan suatu proses dari identifikasi, pengelompokan, pengorganisasian, dan menghasilkan informasi yang relevan pada sebuah domain berdasarkan pengkajian sistem yang ada dan sejarah perkembangannya, juga pengetahuan yang diperoleh, serta pengetahuan dari teori dan teknologi yang akan diterapkan pada pengembangan sistem yang dimaksud.

Tujuan utama dari analisa berorientasi obyek adalah memodelkan sistem yang nyata dengan penekanan apa yang harus dilakukan oleh sistem.

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

Maka landasan teori diagram-diagram UML yang menjadi alat bantu pada tahap analisa berorientasi obyek yaitu :

1) Activity Diagram

Activity diagram menggambarkan berbagai alur kerja dari satu aktifitas ke aktifitas lainnya dalam sistem yang sedang dirancang, bagaimana masing-masing alur aktifitas berawal, decision yang mungkin terjadi, dan bagaimana akhir dari kegiatan mereka. *Activity diagram* dipakai pada *business modelling* untuk memperlihatkan urutan aktifitas proses bisnis.

Simbol-simbol yang digunakan pada saat pembuatan *activitydiagram* adalah sebagai berikut :

a) *Start point* / *Initial Node*, diletakkan pada pojok kiri atas dan menggambarkan awal terjadinya aktifitas.

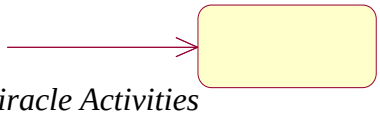
b) *End Point* / *Activity Final Node*, bisa diletakkan dimana saja dan menggambarkan berakhirnya aktifitas

c) *Activities*, menggambarkan suatu proses/kegiatan yang lebih dikenal dengan activity state.

Jenis –jenis *activity state* :

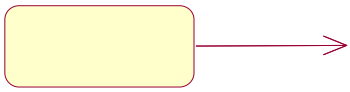
(1) Black Hole Activities

Ada masukan dan tidak ada keluaran. Biasanya digunakan jika dikehendaki ada 1 atau lebih transisi.



(2) *Miracle Activities*

Tidak ada masukan dan ada keluaran. Biasanya dipakai pada waktu start point dan dikehendaki ada 1 atau lebih transisi.

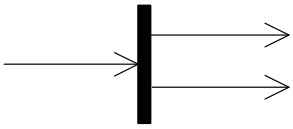


(3) *Parallel Activities*

Suatu activity yang berjalan secara berbarengan yang terdiri dari :

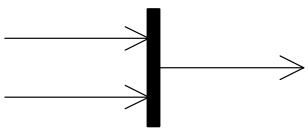
(a) *Fork* (percabangan)

Mempunyai 1 transisi masuk dan 2 atau lebih transisi keluar.



(b) *Join* (penggabungan)

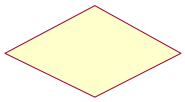
Mempunyai 2 atau lebih transisi masuk dan hanya 1 transisi keluar. Fork harus berhubungan dengan join.



(4) *Decision Point*

Digambarkan dengan lambang belah ketupat, mempunyai transisi (sebuah garis dari/ ke *decision point*). Setiap transisi

yang ada harus mempunyai *guard* (kunci) dan tidak ada sebuah keterangan (pertanyaan) pada tengah belah ketupat seperti pada *flowchart.*



(5) Guard

Sebuah kondisi benar sewaktu melewati sebuah transisi dan digambarkan dengan diletakkan diantara tanda [ ]. Setiap transisi dari /ke *decision point* harus mempunyai guard. Gunakan [*otherwise*] untuk menangkap suatu kondisi yang belum terdeteksi.

[.....]

(6) Swimlane

Sebuah cara untuk mengelompokkan *activity* berdasarkan *actor* (mengelompokkan *activity* dalam sebuah urutan yang sama). *Actor* bisa dituliskan nama *actor* ataupun sekaligus dengan lambang *actor* pada *usecase diagram.*

(7) Transition

Menggambarkan aliran perpindahan kontrol antara *state.*

2) Analisa Dokumen Keluaran

Analisa dokumen keluaran adalah analisa yang menggunakan keluaran-keluaran yang berbentuk informasi atau laporan-laporan yang dihasilkan oleh proses yang ada dalam sistem. Keluaran dapat merupakan masukan untuk subsistem yang lain. Misalnya untuk sistem komputer, panas yang dihasilkan adalah keluaran yang tidak berguna dan merupakan hasil sisa pembuangan. Sedangkan informasi adalah keluaran yang dibutuhkan.

3) Analisa Dokumen Masukan

Analisa dokumen masukan adalah energi yang dimasukkan ke dalam sistem, yang dapat berupa masukan perawatan (maintenance input) dan masukan sinyal (signal input). Masukan perawatan adalah energi yang dimasukkan supaya sistem dapat beroperasi, sedangkan masukan sinyal adalah energi yang diproses untuk mendapatkan keluaran. Sebagai contoh di dalam sistem komputer, program adalah *maintenance input* yang digunakan untuk mengoperasikan komputer dan data adalah signal input untuk diolah menjadi informasi

4) Use Case Diagram

Use Case Diagram adalah deskripsi fungsi dari sebuah sistem yang dilihat dari sudut pandang pengguna. *Use Case* bekerja dengan cara mendeskripsikan tipikal interaksi antar *user* (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Urutan langkah-langkah yang menerangkan antara pengguna dan sistem disebut *scenario*. Dengan demikain *use case* bisa dikatakan sebagai serangkaian *scenario* yang digabungkan bersama-sama oleh tujuan umum pengguna. Sedangkan pengguna itu sendiri didalam *use case* disebut dengan *actor*.

Actor adalah sebuah peran yang bisa dimainkan oleh pengguna dalam interaksinya dengan sistem. *Model Use Case* adalah bagian dari

model *reqruitment* (Jacobson et all, 1992). *Use case* sendiri dapat difungsikan untuk memberikan spesifikasi fungsi-fungsi yang ditawarkan oleh sistem dari perspektif *user*, dalam hal ini untuk menggambarkan kebutuhan sistem dari sudut pandang *user* (pengguna). Selain itu pula *use case* diagram lebih memfokuskan pada proses komputerisasi dalam sebuah sistem dan menggambarkan interaksi yang terjadi antara *actor* dengan *use case*.

Secara umum *use case diagram* terdiri dari :

a) *Actor*

Actor adalah sebuah peran yang bisa dimainkan oleh pengguna dalam interaksinya dengan sistem. Untuk mengidentifikasikan *actor* harus ditentukan pembagian kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. *Actor* dilukiskan dengan peran yang mereka mainkan dalam *use case*, seperti staf penjualan, pelanggan, dll.

b) *Use case*

Use case menggambarkan perilaku, termasuk didalamnya interaksi antara *actor* dengan sistem. *Use case* dibuat berdasarkan keperluan *actor*, merupakan “apa” yang dikerjakan sistem bukan “bagaimana” sistem mengerjakannya. Setiap *use case* harus diberi nama yang menyatakan apa hal yang dicapai dari hasil interaksinya dengan *actor*. Nama *use case* boleh terdiri dari beberapa kata dan tidak boleh ada dua *use case* yang memiliki nama yang sama.

c) *Relationship* (relasi) / *Association* (Asosiasi)

Asosiasi menggambarkan aliran data / informasi. Asosiasi / relasi juga digunakan untuk menggambarkan bagaimana *actor* terlibat dalam *use case.* Relasi (*relationship)* digambarkan sebagai bentuk garis antara dua simbol dalam *use case diagram.*

Ada empat jenis relasi / asosiasi yang dapat timbul pada *use case diagram,* yaitu :

(1) *Asosiasi* 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.

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

(2) *Asosiasi* antara *use case*

Relasi antara *use case* dengan *use case* :

1. *Include,* menggambarkan suatu *use case* termasuk di dalam *use case* lain (diharuskan). Contohnya adalah pemanggilan sebuah fungsi program.

   Digambarkan dengan garis lurus berpanah dengan tulisan *<<include>>*.

2. *Extend,* digunakan ketika hendak menggambarkan variasi pada kondisi perilaku normal dan menggunakan lebih banyak control *form* dan mendeklarasikan ekstensi pada *use case* utama atau dengan kata lain adalah perluasan dari *use case* lain jika syarat atau kondisi terpenuhi.

   Digambarkan dengan garis lurus berpanah tertutup dengan tulisan *<<extend>>*.

3. Generalization / Inheritance, digambarkan dengan garis lurus berpanah tertutup dari *base use case* ke *parent use case.*

(3) *Generalization / Inheritance* antar *use case*

Generalization dipakai ketika ada sebuah perlakuan khusus *(single condition)* dan merupakan pola hubungan *base-parent use case*. Digambarkan dengan *generalization / Inheritance* antar *use case* secara vertikal dengan *inheriting use case* dibawah *base-parent use case.*

(4) *Generalization / Inheritance antar actors* .

Digambarkan *generalization* antar *actors* secara *vertical* dengan *inheriting actor* dibawah *base / parent use case.*

5) Deskripsi Usecase Diagram

Menjelaskan setiap *use case* yang digunakan dalam sistem yang diusulkan. Deskripsi *use case* terdiri dari:

a) *Use case*, adalah nama kegiatan yang dilakukan dalam sistem.
b) *Actor*, merupakan orang yang melakukan kegiatan dalam sistem.
c) *Deskripsi*, menjelaskan kegiatan yang dilakukan *actor* dalam sistem.

c. Perancangan Berorientsi Obyek

Menurut [Ariesto Hadi Sutopo 2002] "*Object Oriented Design* merupakan tahap lanjutan setalah analisis berorientasi obyek dimana tujuan sistem diorganisasi ke dalam sub-sistem berdasarkan struktur analisis dan arsitektur yang dibutuhkan". Sedangkan menurut [Jeffery L.Whitten et al 2004] "Perancangan berorientasi obyek merupakan tahap lanjutan setelah analisa berorientasi obyek, dengan pendekatan yang digunakan untuk mengsfesifikasi kebutuhan-kebutuhan sistem dengan mengkolaborasi obyek-obyek, atribut-atribut, dan metode-metode yang ada.

1) ERD

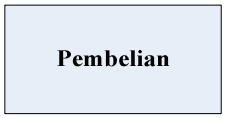
Entity Diagram Relationship (ERD) menggambarkan hubungan antar data yang ada dan tidak menggambarkan proses-proses yang terjadi. Bagi perancang basis data, ERD berguna untuk memodelkan sistem yang nantinya akan dikembangkan basis datanya. Sebuah diagram ER tersusun atas beberapa komponen, yaitu entitas, atribut, relasi, garis dan *cardinality* (tingkat hubungan).

a) Entitas

Merupakan obyek dasar yang terkait dalam sistem. Obyek tersebut dapat berupa orang, benda atau hal yang keterangannya perlu disimpandi dalam basis data. Untuk menggambarkan entitas dilakukan dengan mengikuti aturan sebagai berikut :

(1) Entitas dinyatakan dengan simbol persegi panjang

(2) Nama entitas dituliskan di dalam simbol dalam simbol persegi panjang. Contoh :



(3) Nama entitas berupa kata benda

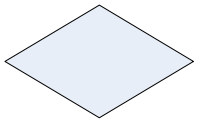
(4) Nama entitas sedapat mungkin menggunakan nama yang mudah dipahami dan dapat menyatakan maknanya dengan jelas.

b) Relationship

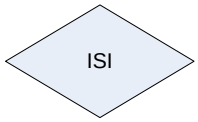
Merupakan kejadian yang menggambarkan hubungan antara dua atau lebih entitas yang keterangannya perlu disimpan di dalam

basis data. Aturan menggambarkan *relationship* adalah sebagai berikut :

(1) *Relationship* digambarkan dengan belah ketupat



(2) Nama *relationship* dituliskan di dalam simbol belah ketupat



(3) *Relationship* menghubungkan dua entitas

(4) Nama *relationship* berupa kata kerja dasar

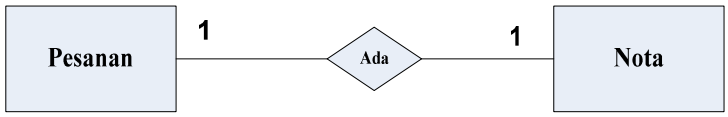
(5) Nama *reletionship* harus mudah dipahami

c) *Cardinality* (Tingkat hubungan)

Angka yang menunjukkan beberapa *entity* dari sebuah *entity set* akan berhubungan dengan beberapa *entity* pada *entity set* yang lain. Tingkat *cardinality* dapat dikelompokkan dalam 3 jenis yaitu :

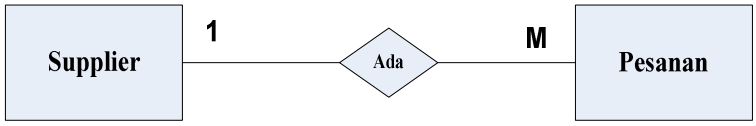
(1) *One to one* (1 : 1)

Terjadi jika kejadian atau transaksi di antara dua entitas yang berhubungan hanya memungkinkan terjadi sebuah kejadian atau transaksi pada kedua entitas. Secara lebih teknis, jika nilai yang digunakan sebagai penghubung pada entitas pertama hanya dimungkinkan muncul satu kali saja pada entitas kedua yang saling berhubungan.



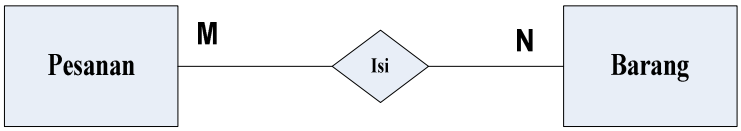
(2) *One to Many* (1 : M) atau *Many to One* (M : 1)

Terjadi jika kejadian atau transaksi diantara dua entitas yang berhubungan hanya memungkinkan terjadi satu kali dalam entitas pertama dan hanya dapat terjadi lebih dari satu kejadian pada entitas kedua. Begitu juga sebaliknya, jika kondisi yang terjadi adalah *many to one*.



(3) *Many to Many* (M : N)

Terjadi jika kejadian atau transaksi diantara entitas yang berhubungan memungkinkan terjadi lebih dari satu kali dalam entitas pertama dan entitas kedua. Secara lebih teknis, jika nilai yang digunakan sebagai penghubung pada entitas pertama dimungkinkan muncul lebih dari satu kali, baik pada entitas pertama maupun pada entitas kedua yang saling berhubungan.



(d) Elemen data / Atribut (Atributte)

Merupakan keterangan-keterangan yang terkait dengan sebuah entitas. Atribut juga merupakan karakteristik dari entitas yang berfungsi sebagai penjelas entitas. Atribut di dalam entitas harus ada yang menjadi *key* (kunci). Key adalah salaha satu gabungan dari beberapa atribut yang dapat membedakan semua data (Row) dalam tabel secara unik/tidak unik.

Jenis-jenis key antara lain:

(i). *Primary* adalah Atribut (field) yang mengidentifikasi sebuah *record* dalam entitas (field) dan bersifat unik.

(ii). *Foreign key* adalah Atribut (field) yang bukan *key*, tetapi adalah *primary key* pada entitas (field) yang lain.

(iii). *Candidate key* adalah atribut (field) yang bisa dipilih menjadi *primary key*.

(iv). *Alternate key* adalah *candidate key* yang tidak terpilih menjadi *primary key*.

(v). *Composite key* adalah *primary key* yang dibentuk dari beberapa *field*.

Jika dapat diterapkan dengan benar, maka penggunaan ERD dalam pemodelan data akan memberi keuntungan, bagi perancang maupun pemakai antara lain :

a) Memudahkan perancang dalam hal menganalisis sistem yang akan dikembangkan.
b) Memudahkan perancang saat merancang basis data
c) Rancangan basis data yang dikembangkan berdasarkan ERD umumnya telah berada pda bentuk optimal.
d) Dalam banyak kesempatan, penggunaan simbol-simbol grafis akan lebih mudah dipahami oleh para pemakai.
e) Dengan menggunakan ERD, pemakai umumnya akan mudah memahami sistem dan basis data yang dirancang oleh perancang.

Kelemahan ERD antara lain:

a) Kebutuhan media yang sangat luas
b) Seringkali ERD tampil sangat ruwet

1) Logical Record Structure (LRS)

Logical record structure (LRS) digambarkan dengan kotak empat persegi panjang den dengan memiiki nama yang sangat unik. Beda LRS dengan ERD nama *type record* berada di luar kotak *field type record* ditempatkan. Dua metode yang dapat digunakan, dimulai dengan hubungan kedua model yang dapat dikonversikan ke LRS. Metode yang lain dimulai dengan ERD yang langsung dikonversikan ke LRS.

Dalam kaitannya dengan konversi ke *logical record structure*, untuk perubahan yang terjadi dengan mengikuti aturan-aturan berikut :

(a) Setiap entitas diubah ke bentuk kotak dengan nama entitas, berada diluar kotak dan atribut berada didalam kotak.

(b) Sebuah *relationship* kadang disatukan, dalam sebuah kotak bersama entitas, kadang sebuah kotak bersama-sama dengan entitas, kadang pula disatukan dalam sebuah kotak tersendiri.

2) Tabel / Relasi

Tabel adalah bentuk pernyataan data secara grafis dua dimensi, yang terdiri dari kolom dan baris. *Konversi* dari *logical record structure* menjadi nama relasi dan tiap atribut menjadi sebuah kolom dalam relasi.

3) Spesifikasi Basis Data

Menurut [Sutanta 2004] “Sistem basis data dapat didefinisikan sekumpulan subsistem yang terdiri atas basis data dengan para pemakai yang menggunakan basis data secara bersama-sama, personal-personal yang merancang basis data, serta sistem komputer untuk mendukungnya”.

Dengan memahami pengertian tersebut di atas, maka istilah basis data dapat dipahami sebagai suatu kumpulan data terhubung (interrelated data) yang disimpan secara bersama-sama pada suatu media, tidak perlu suatu kerangkapan data (kalaupun ada maka kerangkapan data tersebut harus seminimal mungkin dan terkontrol (*controlled redudancy*), data disimpan dengan cara-cara tertentu sehingga mudah digunakan atau ditampilkan kembali, data dapat digunakan oleh satu atau lebih program-program aplikasi secara optimal, data disimpan tanpa mengalami ketergantungan dengan program yang akan menggunakannya, data disimpan sedemikian rupa sehingga prosees penambahan, pengambilan dan modifikasi data dapat dilakukan dengan mudah dan terkontrol.

4) Rancangan Dokumen Keluaran

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

5) Rancangan Dokumen Masukan

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

6) Rancangan Layar Program

Rancangan layar program merupakan bentuk tampilan yang bila dijalankan pada program tersebut, maka sistem akan menampilkan rancangan pada layar komputer dimana sebagai sarana antar muka dengan pemakai yang akan dihasilkan dari sistem yang dirancang.

7) Sequence Diagram

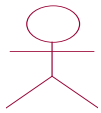
Sequence diagram adalah *visual coding* (perancangan form/layar). *Sequence diagram* menggambarkan interaksi antar obyek di dalam dan di sekitar sistem (termasuk pengguna, *display*

dan sebagainya) berupa *message* yang digambarkan terhadap waktu. 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*.

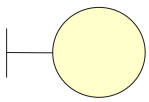
[Jeffrey L. Whitten et al 2004] mengungkapkan "suatu diagram UML yang memodelkan logika dari suatu *use case* dengan menggambarkan interaksi berupa pengiriman pesan (messsage) antar obyek dalam urutan waktu".

Beberapa simbol yang umum digunakan pada *sequence diagram*, yaitu :

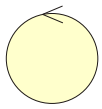
a) *Actor*, menggambarkan seseorang atau sesuatu yang berinteraksi dengan sistem.



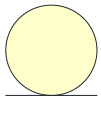
b) *Boundary*, sebuah objek yang menjadi penghubung antara *user* dengan sistem.



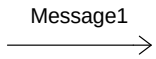
c) *Control*, menggambarkan perilaku, mengatur, mengkoordinir perilaku sistem dan dinamika dari suatu sistem dan mengontrol alur kerja suatu sistem.



d) *Entity*, menggambarkan informasi yang harus disimpan oleh sistem.



e) *Object Message*, menggambarkan pesan/hubungan antar objek yang menunjukkan urutan kejadian yang terjadi.

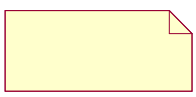


f)*Message to self*, sebuah obyek yang mempunyai sebuah *operation* kepada dirinya sendiri.

g) *Activation*, activation mewakili sebuah eksekusi operasi dari obyek, panjang kotak ini berbanding lurus dengan durasi aktivasi sebuah operasi.

h) *Lifeline*, garis titik-titik yang terhubung dengan obyek, sepanjang *lifeline* terdapat *activation*.

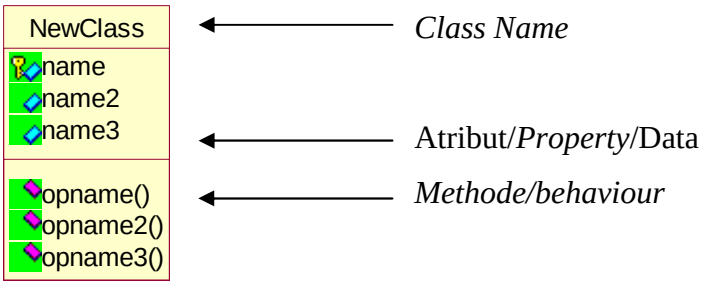
i)*Loop*, menggambarkan dari suatu kejadian yang dilakukan secara berulang-ulang.



8) 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/property) suatu sistem, sekaligus menawarkan layanan untuk memanipulsi keadaan tersebut (metoda/fungsi). *Class diagram* menggambarkan struktur objek sistem, dimana diperlihatkan hubungan antara mereka.

Class diagram memiliki tiga area pokok yaitu nama, atribut, *method*, ditambah lagi asosiasi sabagai penghubung.



a) *Class name*, merupakan nama dari sebuah *class*.

b) Atribut

Atribut adalah sebuah nilai data yang dimiliki oleh *class*. Nama, umur, berat badan adalah atribut dari obyek manusia atau orang. Setiap atribut memiliki nilai untuk obyek *instance*.

c) Method

Method adalah sesuatu yang bisa dilakukan oleh kelas atau implementasi dari sebuah operasi ke dalam sebuah kelas.

d) Asosiasi (*Association*)

Asosiasi adalah *class-class* yang terhubungkan secara konseptual. Setiap asosiasi mempunyai dua *association end.* Sebuah *association end* juga memiliki "*multiplicity*" yang menunjukkan beberapa banyak objek yang berpartisipasi dalam

satu relasi. *Multiplicity* menunjukkan batasan terendah dan tertinggi untuk objek-objek yang berpartisipasi. *Multiplicity* yang paling umum digunakan adalah 1, *, dan 0..1.

Hubungan antar *Class* antara lain :

(1) 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*.

(2) Agregasi, yaitu hubungan yang menyatakan bagian ("terdiri atas..").

(3) 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.

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

3. Teori Pendukung

Sistem pembelian digunakan dalam perusahaan dalam pengadaan barang yang diperlukan oleh perusahaan. Transaksi pembelian dibedakan menjadi dua yaitu pembelian lokal dan pembelian impor. Pembelian lokal adalah pembelian dari pemasok dalam negeri, pembelian impor adalah pembelian dari pemasok luar negeri. Pembelian tunai adalah pembelian yang dilakukan oleh perusahaan dengan mengeluarkan kas untuk pembayaran barang yang dibeli untuk aktivitas perusahaan dan untuk barang persediaan.