BAB II
LANDASAN TEORI

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

2.1. Konsep Sistem Informasi

2.1.1. Konsep Dasar Sistem dan Informasi

1) Konsep Dasar Sistem

Suatu system pada dasarnya adalah sekelompok unsure yang erat hubungannya dengan yang lain yang berfungsi bersama-sama untuk mencapai tujuan tertentu. Eriyanto, 1999 dalam bukunya menyatakan bahwa :

" Sistem dapat didefinisikan suatu kesatuan yang terdiri dari komponen atau elemen yang dihubungkan bersama untuk memudahkan aliran informasi, materi atau energy untuk mencapai suatu tujuan. Istilah ini sering dipergunakan untuk menggambarkan suatu set entitas yang berinteraksi, di mana suatu model matematika seringkali bias dibuat."

2) Konsep Dasar Informasi

Menurut Aji Supriyanto (2005:190) mengungkapkan bahwa informasi adalah :

" Data yang merupakan fakta yang tercatat dan selanjutnya dilakukan pengolahan (proses) menjadi bentuk yang berguna atau bermanfaat bagi pemakainya."

Untuk dapat berguna, maka informasi harus didukung oleh tiga pilar, yaitu seebagai berikut :

a. Akurat (*Accurate*)

Akurat berarti informasi harus bebas dari kesalahan-kesalahan dan tidak menyesatkan. Akurat juga berarti informasi harus jelas mencerminkan maksudnya. Informasi harus akurat karena dari sumber informasi

sampai ke penerima informasi kemungkinan banyak terjadi ganggua (noise) yang dapat merubah atau merusak informasi tersebut.

b. Tepat pada waktunya (*TimeLiness*)

Tepat pada waktunya berarti informasi yang datang pada penerima tidak boleh terlambat. Informasi yang sudah using tidak akan mempunyai nilai lagi karena informasi merupakan landasan didalam pengambilan keputusan.

c. Relevan (*Relevance*)

Relevan berarti informasi tersebut mempunyai manfaat untuk pemakainya. Relevansi informasi untuk tiap-tiap orang berbeda. Nilai informasi bagi seorang pemakai ditentukan oleh keandalan (reliabilitas).

Keluaran yang didukung oleh ketiga pilar ini tidak dapat dikatakan sebagai informasi yang berguna, tetapi merupakan sampah (garbage).

Aji Supriyanto (2005:190) mengungkapkan bahwa data meupakan :

" Fakta atau nilai (value) yang tercatat atau merepresentasikan deskripsi dari suatu obyek.

2.1.2. Konsep Dasar Sistem Informasi

Menurut James A. O'Brien (2007:45) mendefinisikan system informasi merupakan :

" Gabungan yang terorganisasi dari manusia, perangkat lunak, perangkat keras, jaringan komunikasi dan sumber data dalam mengumpulkan, mengubah, dan menyebarkan informasi dalam organisasi."

2.2. Analisa dan Perancangan Berorientasi Obyek dengan UML

a. UML (*Unified Modeling Language*)

1) Pengertian UML

Menurut David M. Kroenke, Database Processing Jilid I edisi 9, halaman 60, Erlangga mengungkapkan bahwa UML (Unified Modeling Language) merupakan :

" Unified Modeling Language merupakan himpunan struktur dan teknik untuk pemodelan desain program berorientasi objek (OOP) serta aplikasinya."

Tujuan utama UML diantaranya adalah untuk :

a) Memberikan model yang siap pakai, bahasa pemodelan visual yang ekspresif untuk mengembangkan dan saling menukar model dengan mudah dan dimengerti secara umum.
b) Memberikan bahasa pemodelan yang bebas dari berbagai bahasa pemograman dan proses rekayasa.
c) Menyatukan praktek-praktek terbaik yang terdapat dalam pemodelan.

2) Sejarah UML

Menurut Martin Fowler (2005:201) mengungkapkan bahwa sejarah Unified Modeling Language adalah sebagai berikut :

a) Pada Oktober 1994, Dr. James Rimbaugh bergabung dengan perusahaan rational software, dimana Grady Booch sudah bekerja di sana sebelumnya. Grady Booch mengembangkan Object Oriented Design (OOD) dan Dr. James Rimbaugh mengembangkan Object Modelling Technique (OMT). Duet mereka pada Oktober 1995 menghasilkan Unified Method versi 0.8.

b) Banyak perusahaan software merasakan bagaimana pentingnya UML dalam tujuan strategis mereka, sehingga beberapa perusahaan

membentuk sebuah konsorsium yang terdiri dari perusahaan-perusahaan seperti :

(1) Microsoft

(2) Oracle

(3) IBM

(4) Hewlett-Packard

(5) Intellicorp

(6) I-Logix

(7) DEC, Digital Equipment

(8) Texas Instrument

(9) Rational Software

(10) Icon Computing

(11) MCI Systemhouse

(12) Unisy Platinum technology

(13) Ptech

(14) Taskon and Reich Technologies

(15) Soft Team

c) Pada bulan September 1997 lahirnya UML versi 1.1 dengan 8 buah diagram yaitu :

(1) Use Case Diagram

(2) Activity Diagram

(3) Sequence Diagram

(4) Collaboration Diagram

(5) Class Diagram

(6) Statechart Diagram

(7) Component Diagram

(8) Deployment Diagram

d) OMG didirikan pada tahun April 1989 oleh sebelas perusahaan software, dengan kantor pusat di needham, MA, USA.

e) Pada tahun 1999 lahirlah UML versi 1.3 menjadi 9 buah diagram dengan penambah object diagram.

f) Pada tahun 2002 lahirlah UML versi 2.0 menjadi 13 buah diagram, dengan pemambahan dan penggantian yaitu :

(1) Use Case Diagram

(2) Activity Diagram

(3) Sequence Diagram

(4) Communication Diagram (Collaboration diagram in versi 1.x)

(5) Class Diagram

(6) State Machine Diagram (Statechart Diagram in versi 1.x)

(7) Component Diagram

(8) Deployment Diagram

(9) Composite Structure Diagram

(10) Interaction Overview Diagram

(11) Object Diagram

(12) Package Diagram

(13) Timing Diagram

g) Modelling with UML versi 2.0 pemodelan dengan UML ada 13 diagram yang terbagi menjadi 3 kategori yaitu :

(1) Structure Diagram

Menggambarkan elemen dari spesifikasi yang mengabaikan waktu (time).

(a) Class Diagram

(b) Object Diagram

(c) Component Diagram

(d) Deployment Diagram

(e) Composite Diagram

(f) Package Diagram

(2) Behavior Diagram

Menggambarkan ciri-ciri behavior/method/function dari sebuah atau business process.

(a) *Use Case Diagram*

(b) *Activity Diagram*

(c) *State Machine Diagram*

(3) Interaction Diagram

Bagian dari behavior diagram yang menggambarkan object interactions.

(a) *Communication*

(b) *Interaction Overview*

(c) *Sequence*

(d) *Timing*

b. Analisa Sistem Berorientasi Obyek

Analisa system adalah suatu proses untuk mengindentifikasi dan mengevaluasi permasalahan-permasalahan, penyebab-penyebab masalah, mengidentifikasi kebutuhan-kebutuhan system yang kembangkan (Adi Nugroho, 2005: 9).

Melakukan kajian dan menemukan berbagai faktor dari prosedur penyelenggaraan pengolahan data yang berlangsung saamiagrt ini (present system) untuk bisa memenuhi kebutuhan akan informasi yang efektif, itulah yang menjadi titik berat dari sebuah proses penganalisa akan sebuah system yang akan dikomputerisasi.

Keberhasilan dari tahap analisa adalah memahami kebutuhan-kebutuhan system dan membuat konsep system baru yang menggambarkan apa yang harus dilakukan system guna memenuhi kebutuhan-kebutuhan system.

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

Pada tahap analisa berorientasi obyek, obyek bisnis dalam sebuah system didefinisikan seperti siapa atau apa aktornya dan bagaimana mereka berkerjasama dalam aplikasi. Dalam hal ini penulis

menggunakan Use Case untuk mengidentifikasikan apa yang akan pengguna kerjakan dengan system atau perangkat lunak yang akan dikembangkan. Dan mengidentifikasikan actor termasuk didalamnya adalah siapa yang akan menggunakan system.

Mengembangkan proses bisnis sederhana yang memang terjadi di organisasi yang sedang dianalisis dengan membuat *activity diagram.*

Dari penjelasan di atas, penulis menjabarkan landasan teori diagram-diagram UML yang menjadi alat bantu ada tahap analisis berorentasi obyek(*OOA-Object Oriented Analysis*).

1) Activity Diagram

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

Dengan kata lain, *activity diagram* adalah teknik untuk mendeskripsikan logika procedural, prose 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 parallel sedangkan *flowchart* tidak bisa. (Munawar, 2005:109).

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

a) *Start Point (initial node)* dengan tanda

Simbol 3.1 *Start Point*

b) *End Point (activity final node)* dengan tanda

Simbol 3.2 *End point*

c) *Activities*

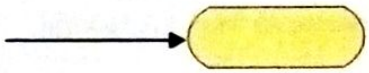
Menggambarkan proses bisnis dan dikenal sebagai *activity state*. Digambarkan dengan bentuk

Simbol 3.3 *Activities*

Jenis *activities*

(1) *Black Hole Activities*

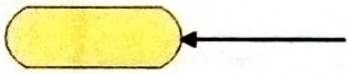
Ada masukan dan tidak ada keluaran, biasanya digunakan jika dikehendaki ada 1 atau lebih transisi.



Simbol 3.4 *Black Hole Activities*

(2) *Miracle Activities*

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



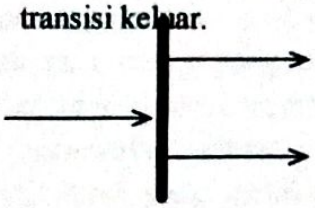
Simbol 3.5 *Miracle Activities*

(3) *Paraller Activities*

Suatu *activity* yang berjalan secara berbarengan terdiri dari :

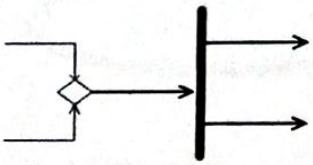
a. *Fork* (Percabangan)

Mempunyai 1 transisi masuk dan 2 atau lebih transisi keluar.



Simbol 3.6 *Fork* (Percabangan)

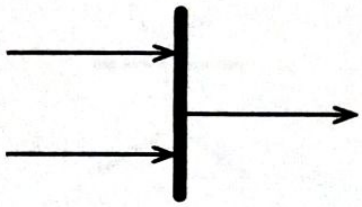
Ketika ada >1 transisi masuk ke *fork* yang sama, gabungkan dengan sebuah *decision point*



Simbol 3.7 *Fork Decision point*

b. *Join* (Penggabungan)

Mempunyai 2 atau lebih transisi masuk dan hanya 1 transisi keluar, *fork* harus berhubungan dengan *join.*



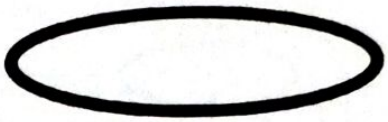
Simbol 3.8 *Join* (Penggabungan)

2) *Use Case Diagram*

Use case diagram digunakan untuk memodelkan bisnis proses berdasarkan perspektif pengguna sistem. *Use case diagram* terdiri atas diagram untuk *use case* dan *actor*. *Actor* merepresentasikan orang yang akan mengoperasikan atau orang yang berinteraksi dengan sistem aplikasi. *Use case* merepresentasikan operasi-operasi yang dilakukan oleh *actor*. *Use case* digambarkan berbentuk elips dengan nama operasi dituliskan didalamnya. *Actor* yang melakukan operasi dihubungkan dengan garis lurus ke *use case*.

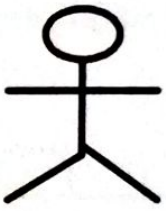
a) *Use Case*

Use case dinotasikan dengan simbol (*horizontal ellipse*)



Simbol 3.9 *Use Case*

b) *Actor* menggambarkan orang, sistem atau eksternal entitas / stakeholder yang menyediakan atau menerima informasi dari system. Actor digambarkan dengan simbol *stick figure* atau dengan gambar visual.



Simbol 3.10 *Actor*

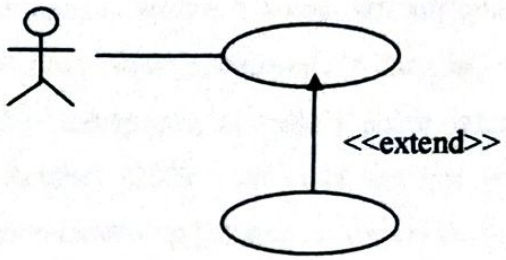
c) *Association* adalah abtraksi berupa garis tanpa panah yang menghubung antara aktor dan *use case*.

Simbol 3.11 *Association*

d) *Include* Menunjukkan bahwa suatu use case seluruhnya merupakan fungsionalitas dari use case lainya.

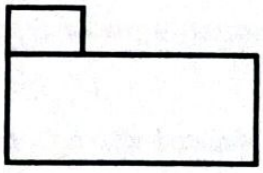
Simbol 3.12 *Include*

e) *Extend* menunjukkan suatu *use case* merupakan tambahan fungsional dari *use case* lainnya jika suatu kondisi terpenuhi.



Simbol 3.13 *Extend*

f) *Packages* digambarkan sebagai sebuah direktori yang berisikan model – model elemen. *Packages* digunakan untuk mengorganisasikan sebuah diagram yang besar menjadi beberapa diagram kecil.



Simbol 3.14 *Packages*

3) ERD (*Entity Relationship Diagram*)

Menurut pendapat Kronke (2006 : 37-40) *Entity Relationship Diagram (ERD)* adalah adalah suatu pemodelan konseptual yang didesain secara khusus untuk mengidentifikasikan entitas yang menjelaskan data dan hubungan antar data, yaitu dengan menuliskan dalam *cardinality*. Elemen – elemen yang membentuk ERD adalah :

1. *Entity* yaitu suatu entitas yang dapat berupa orang, tempat, obyek, atau kejadian yang dianggap penting bagi perusahaan, sehingga segala atributnya harus dicatat dan disimpan dalam basis data. Contoh dari *entity* adalah *employee*, *customer*, sales order.
2. *Attribute* yaitu setiap entitas mempunyai karakterisik tertentu yang dinamakan dengan atribut. Contoh dari attribute adalah *Employee Name*, *Customer Name*, *Employee ID*, dan *Customer ID*.
3. *Relationship* merupakan hubungan suatu jalinan antara entitas. Menurut Romney (2009 : 596) ada tiga tipe *relationship*, yaitu : *One-to-one relationship* Dimana maximum *cardinality* setiap *entity* adalah 1. Contoh : Satu nasabah bank hanya memiliki satu account. *One-to-many relationship* (1:N). Dimana maximum *cardinality* dari suatu *entity* adalah 1 dan *maximum cardinality* dari entity lain adalah N. Contoh : Satu nasabah bank dapat memiliki lebih dari satu *account*. *Many-to-many relationship* (M:N). Dimana *maximum cardinality* kedua *entity* yang berhubungan adalah N. Contoh : Satu nasabah dapat memilliki beberapa *account* dan satu *account* dapat dimiliki oleh beberapa nasabah (rekening bersama).

4. *Cardinality* merupakan kendala-kendala yang timbul dalam hubungan antar entitas. *Cardinality* seringkali diekspresikan dalam sepasang angka. Angka pertama disebut *minimum cardinality* dan angka kedua disebut *maximum cardinality* (Romney, 2009). *Minimum Cardinality* mengindikasikan angka terkecil dari baris yang dapat dihubungkan dalam *relationship*. *Minimum cardinality*

bisa 0 atau 1. Yang dimaksud dengan *minimum cardinality* 0 adalah setiap baris entity pada *relationship* lain.

Sedangkan minimum *cardinality* 1 menunjukkan bahwa setiap baris dari entity harus dihubungkan dengan paling sedikit satu baris dari *entity* lain. *Maximum Cardinality* mengindikasikan angka terbesar dari baris yang dapat dihubungkan dalam relationship. *Maximum Cardinality* bisa 1 atau N. Simbol yang ada menunjukkan setiap baris dalam tabel dapat dihubungkan dengan beberapa baris pada tabel lain. *Maximum Cardinality* 1 menunjukkan bahwa baris dari entity dapat dihubungkan ke paling banyak satu baris dari *entity* lain. Sedangkan minimum cardinality N menunjukkan bahwa satu baris dari *entity* dapat dihubungkan dengan lebih dari satu baris dari *entity* lain.

Simbol-simbol/Notasi yang digunakan dalam *ERD* :

1. Entitas / *Entity* yaitu menggambarkan himpunan orang, tempat, objek dan sebagainya yang berperan dalam sistem.

Simbol 3.25 Entitas

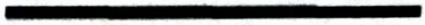
2. Relasi yaitu menggambarkan hubungan yang ada diantara himpunan entitas.

Simbol 3.26 Relasi

3. *Attribute* adalah elemen data yang dimiliki sebuah entitas. Atribut berfungsi mendeskripsikan karakteristik entitas (atribut yang berfungsi sebagai *key* diberi garis bawah)

Simbol 3.27 *Attribute*

4. Garis yaitu sebagai penghubung antara relasi dengan entitas, relasi dan entitas dengan atribut.



Simbol 3.28 Garis Penghubung

Cardinality adalah tingkat hubungan atau derajat relasi.

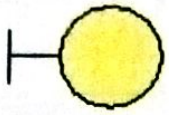
4) *Sequence Diagram*

Sequence diagram menjelaskan secara detil urutan proses yang dilakukan dalam sistem untuk mencapai tujuan dari *use case.* Interaksi yang terjadi antar *class*, operasi apa saja yang terlibat, urutan antar operasi, dan informasi yang diperlukan oleh masing-masing operasi.

1. *Actor* dalam *use case* digambarkan dengan *stick figure* sebagai berikut :

Simbol 3.15 *Actor*

2. *User Interface Class* (UI)

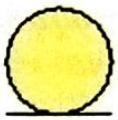


Simbol 3.16 *User Interface Class*

3. *Controller Class* digambarkan sebagai berikut

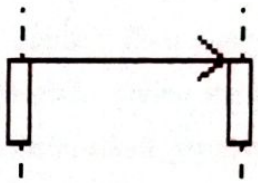
Simbol 3.17 *Controller Class*

4. *Entity* digambarkan sebagai berikut



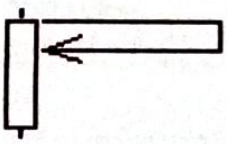
Simbol 3.18 *Entity*

5. *Message* digambarkan dengan garis berpanah terbuka yang menunjukkan arah *message*



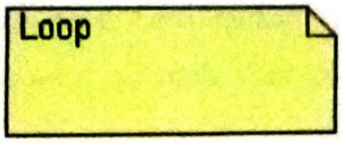
Simbol 3.19 *Message*

6. *Boxes* atau *message* yang dikirim untuk dirinya sendiri digambarkan dengan bentuk.



Simbol 3.20 *Boxes*

7. *Looping logic* yaitu digambarkan dengan sebuah *frame* dengan label *loop* dan sebuah kalimat yang mengindikasikan pengulangan dan *interaction operator loop.*

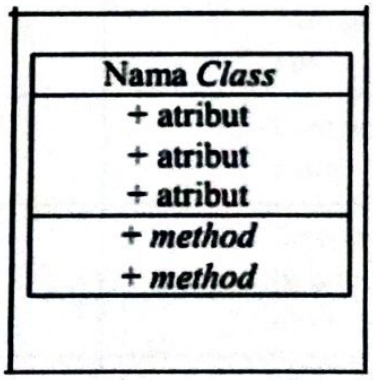


Simbol 3.21 *Looping Logic*

5) *Class Diagram*

Class diagram merupakan diagram yang selalu ada di permodelan sistem berorientasi objek. *Class* diagram menunjukkan hubungan antar *class* dalam sistem yang sedang dibangun dan bagaimana mereka saling berkolaborasi untuk mencapai suatu tujuan. Elemen – eleman *class* diagram dalam pemodelan UML terdiri dari: *Class-class*, struktur *class*, sifat *class* (*class behavior*), perkumpulan/gabungan (*association*), pengumpulan/kesatuan (*agregation*), ketergantungan (*dependency*), relasi-relasi turunannya, keberagaman dan indikator navigasi, dan rolename (peranan/tugasnama). Simbol-simbol class diagram :

1. *Class* adalah blok – blok pembangun pada pemrograman berorientasi obyek. Sebuah c*lass* digambarkan sebagai sebuah kotak yang terbagi atas 3 bagian. Bagian atas adalah bagian nama dari *class*. Bagian tengah mendefinisikan *property*/atribut *class*. Bagian akhir mendefinisikan method – method dari sebuah *class*.



Simbol 3.22 *Class*

2. *Association* yaitu sebuah asosiasi merupakan sebuah *relationship* paling umum antara 2 *class* dan dilambangkan oleh sebuah garis yang menghubungkan antara 2 *class*. Garis ini bisa melambangkan tipe-tipe *relationship* dan juga dapat menampilkan hukum-hukum multiplisitas pada sebuah *relationship*. (Contoh: *One-to-one, one-to-many, many-to-many*).

Simbol 3.23 *Association*

3. *Dependency* yaitu kadangkala sebuah *class* menggunakan *class* yang lain. Hal ini disebut *dependency*. Umumnya penggunaan *dependency* digunakan untuk menunjukkan operasi pada suatu class yang menggunakan *class* yang lain. Sebuah *dependency* dilambangkan sebagai sebuah panah bertitik-titik.

Simbol 3.24 *Dependency*

4. *Multiplicity Indicator*

Indicator	Meaning	Example
0..1	Zero or one	
0..*	Zero or more	
0..n	Zero to n (where n>1)	0..3
1	One only	
1..*	One or more	

1..n	One to n (where n>1)	1..5
*	Many	
N	Only n (where n>1)	9
n..*	n or more, where n >1	7..*
n..m	Where n & m both >1	3..10

Gambar 3.1 *Multiplicity Indicator*