BAB II

TINJAUAN PUSTAKA

II.1. Pengertian Sistem

Untuk mengawali pembahasan tentang analisis dan perancangan sistem informasi, pemahaman akan sistem terlebih dahulu harus ditekankan. Definisi sistem berkembang sesuai dengan konteks di mana pengertian sistem itu digunakan. Berikut akan diberikan beberapa definisi sistem secara umum :

1. Kumpulan dari bagian-bagian yang bekerja sama untuk mencapai tujuan yang sama. Contoh :
   a. Sistem Tatasurya
   b. Sistem Pencernaan
   c. Sistem Transportasi Umum
   d. Sistem Otomatif
   e. Sistem Komputer
   f. Sistem Informasi
2. Sekumpulan objek-objek yang saling berelasi dan berinteraksi serta hubungan antar objek bisa dilihat sebagai satu kesatuan yang dirancang untuk mencapai satu tujuan.

Dengan demikian, secara sederhana sistem dapat diartikan sebagai suatu kumpulan atau himpunan dari unsur atau variabel-variabel yang saling terorganisasi, saling berinteraksi, dan saling bergantung sama lain. *Murdick* dan *Ross* (1993) mendefinisikan sistem sebagai seperangkat elemen yang digabungkan

satu dengan lainnya untuk satu tujuan bersama. Sementara definisi sistem dalam kamus *Webster's Unbriged* adalah elemen-elemen yang saling berhubungan dan membentuk satu kesatuan atau organisasi (Hanif Al Fatta ; 2007 : 3).

II.2. Pengertian Sistem Informasi

Sistem Informasi dapat didefinisikan sebagai kumpulan elemen yang saling berhubungan satu sama lain yang membentuk satu kesatuan untuk mengintegrasikan data, memproses dan menyimpan serta mendistribusikan informasi. Dengan kata lain, SI(Sistem Informasi) merupakan kesatuan elemen-elemen yang saling berinteraksi secara sistematis dan teratur untuk menciptakan dan membentuk aliran informasi yang akan mendukung pembuatan keputusan dan melakukan kontrol terhadap jalannya perusahaan.

Sistem Informasi juga mampu mendukung para pengelola dan staf perusahaan untuk menganalisa permasalahan, memvisualisasikan ikhtisar analisa melalui grafik-grafik dan tabel-tabel, serta memungkinkan terciptanya produk serta layanan yang baru. Sistem Informasi yang baik tentu memiliki sistematika yang jelas, ringkas, dan sederhana. Mulai dari tahap pemasukan data, pengolahan dengan prosedur yang ditentukan, penyajian informasi yang akurat, interprestasi yang tepat dan distribusinya (Budi Sutedjo Dharma Oetomo ; 2006 : 11).

II.3. Sumber Daya Manusia

Menurut Gade (2005:4) dalam bukunya *Teori Akuntansi*, akuntansi adalah istilah yang luas yang menunjukkan teori-teori tertentu, asumsi-asumsi mengenai

cara bertindak, peraturan-peraturan cara mengukur dan prosedur untuk mengumpulkan dan melaporkan informasi yang berguna tentang kegiatan-kegiatan dan tujuan suatu organisasi.

Defenisi spesifik akuntansi adalah ilmu pengetahuan terapan dan seni pencatatan yang dilakukan secara terus menerus menurut sistem tertentu, mengolah dan menganalisis catatan tersebut sehingga dapat disusun suatu laporan keuangan sebagai pertanggungjawaban pimpinan perusahaan atau lembaga terhadap kinerjanya.

Tujuan utama akuntansi adalah memberikan informasi yang diperlukan untuk mengambil putusan bagi manajemen, pemegang saham, pemerintah atau pihak-pihak lain yang berkepentingan sehingga keputusan-keputusan yang benar dapat diambil tentang apa yang sudah terjadi dalam organisasi atau apa yang harus diperbuat dikemudian hari.

II.4. Pengertian Java

Java Language Spesification adalah definisi teknis dari bahasa pemrograman *Java* yang di dalamnya terdapat aturan penulisan sintaks dan semantic *Java.*Referensi lengkap tentang *Java Language Spesification* dapat anda temui pada website resmi *Java*, yaitu http://java.sun.com/docs/books/jl (WahanaKomputer ; 2010 : 3).

Java Memiliki banyak kelebihan, antara lain :

a. Sederhana

   Untuk mempelajari Java tidaklah sulit. Anda bisa mempelajari dan membuat program secara cepat dengan menggunakan Java.

b. Beriorientasi Objek

Meskipun banyak bahasa pemrograman yang menyebut dirinya berorientasi objek, Java bukan merupakan tururan dari bahasa pemrograman manapun dan sama sekali tidak kompatibel dengan *source-code* bahasa apapun.

c. Aman

Banyak bahasa yang pada awalnya dirancang dengan tingkat keamanan yang hampir tidak ada. Fasilitas-fasilitas yang diberikan seringkali dapat dimanfaatkan untuk disusupi oleh berbagai virus. Berbeda dengan Java yang disusun sejak awal dengan prinsip keamanan.

d. Interaktif

Java dirancang memenuhi kebutuhan dunia nyata untuk menciptakan program jaringan yang interaktif.

e. Kokoh

Java membatasi anda dari berbagai hal kunci supaya dapat menemukan kesalahan lebih cepat saat mengembangkan program.

f. Terinterprestasi dan Berkinerja Tinggi

Java dapat diterjemahkan oleh sistem apapun yang memiliki program Java di dalamnya. Java dirancang dengan hati-hati sehingga mudah diterjemahkan ke dalam bahasa asli suatu sistem untuk menghasilkan kinerja yang tinggi.

g. Terdistribusi

Java dapat dengan mudah bekerja dalam lingkungan terdistribusi seperti internet/intranet karena kemampuannya menangani protokol TCP/IP.

h. Dinamis

Karena berorientas objek, Java mudah dalam hal pemeliharaan dan pengembangan, karena kita tidak harus membedah isi program untuk mengubah dan mengembangkan program dengan skala lebih besar(Ridwan Sanjaya; 2005 : 2).

II.5. Pengertian NetBeans

NetBeans merupakan salah satu proyek *open source* yang disponsori oleh *Sun Micro system*. Proyek ini bediri pada tahun 2000 dan telah menghasilkan 2 produk, yaitu NetBeans IDE dan NetBeans Platform. NetBeans IDE merupakan produk yang digunakan untuk melakukan pemrograman baik menulis kode, meng-*compile*, mencari kesalahan dan mendistribusikan program. Sedangkan NetBeans Platform adalah sebuah modul yang merupakan kerangka awal/pondasi dalam bangun aplikasi *desktop* yang besar.

NetBeans juga menyediakan paket yang lengkap dalam pemrograman dari pemrograman standar (aplikasi desktop), pemrograman enterprise, dan pemrograman perangkat mobile. Saat ini NetBeans telah mencapai versi 6.8 (Wahana Komputer ; 2010 : 15).

II.6. Pengertian Database

Database merupakan komponen terpenting dalam pembangunan SI (SistemInformasi), karena menjadi tempat untuk menampung dan mengorganisasikan seluruh data yang ada dalam sistem, sehingga dapat dieksplorasi untuk menyusun-menyusun informasi-informasi dalam berbagai bentuk. *Database* merupakan himpunan kelompok data yang saling berkaitan. Data tersebut diorganisasikan sedemikian rupa agar tidak terjadi duplikasi yang

tidak perlu, sehingga dapat diolah atau di eksplorasi secara cepat dan mudah untuk menghasilkan informasi (Budi Sutedjo Dharma Oetomo: 2006: 99).

II.7. Pengertian MySQL

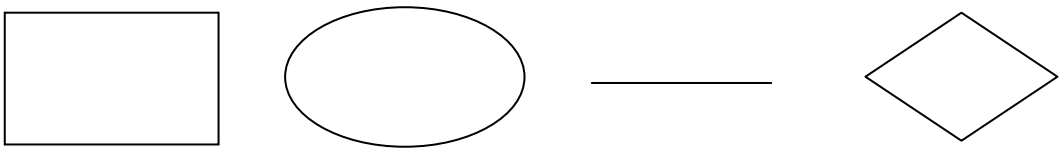
Menurut Supardi (2007:97), perangkat lunak MySQL adalah perangkat lunak basis data server yang terkenal dan bersifat *open-source* dengan dukungan *driver* yang luas dari berbagai *vendor*. MySQL adalah implementasi dari system manajemen basis data relasional (RDBMS) yang didistribusikan secara gratis dibawah lisensi GPL (*General Public License*). Setiap pengguna dapat secara bebas menggunakan MySQL, namun dengan batasan perangkat lunak tersebut tidak boleh dijadikan produk turunan yang bersifat komersial. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam basis data yang telah ada sebelumnya. SQL (*Structured Query Language*). SQL adalah seakuntansi konsep pengoperasian basis data, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis.

Kehandalan suatu system basis data (DBMS) dapat diketahui dari cara kerja pengoptimasi-nya dalam melakukan proses perintah-perintah SQL yang dibuat oleh pengguna maupun program-program aplikasi yang memanfaatkannya. Sebagai peladen basis data, MySQL mendukung operasi basis data transaksional maupun operasi basis data non-transaksional.Pada modus operasi non-transaksional, MySQL dapat dikatakan unggul dalam hal unjuk kerja dibandingkan perangkat lunak pengelolah basis data kompetitorlainnya. Namun demikian pada modus non-transaksional tidak ada jaminan atas reliabilitas

terhadap data yang tersimpan, karenanya modus non-transaksional hanya cocok untuk jenis aplikasi yang tidak membutuhkan reliabilitas data seperti aplikasi blogging berbasis web, CMS, dansejenisnya.Untuk kebutuhan sistem yang ditujukan untuk bisnis sangat disarankan untuk menggunakan modus basis data transaksional, hanya saja sebagai konsekuensinya unjuk kerja MySQL pada modus transaksional tidak secepat unjuk kerja pada modus non-transaksional.

II.8. *EntityRelationship Diagram* (ERD)

EntityRelationship Diagram atau ERD merupakan salah satu alat (tool) berbentuk grafis yang populer untuk *desain database.* Tool ini relatif lebih mudah dibandingkan dengan Normalisasi. Kebanyakan sistem analis memakai alat ini, tetapi yang jadi masalah, kalau kita cermati secara seksama, tool ini mencapai 2NF (Ir. Yuniar Supardi ; 2010 : 448).



Entitas | Atribut/Field | Link (hubungan) | Himpunan Relasi/Interface

Gambar. II.1 Bentuk Simbol ERD
(Sumber : Ir. Yuniar Supardi ; 2010 : 448)

II.9. Kamus Data

Karena DBMS menyimpan kumpulan beberapa *item* data yang terpisah yang dapat digunakan pemakai pada beberapa aplikasi secara bersama-sama, adalah penting bahwa beberapa mekanisme digunakan untuk menyediakan informasi mengenai beberapa *item* data bersangkutan. Itu adalah fungsi dari kamus data.

Kamus data adalah suatufile yang terpisah yang menyimpan informasi seperti :

a. Nama setiap *item*/jenis/kolom data.
b. Struktur data untuk tiap *item*.
c. Program yang menggunakan tiap *item*
d. Tingkat keamanan untuk setiap *item*.

Pemakai yang perlu memperoleh informasi dari database dapat menuju ke kamus data untuk mendapatkan nama dari *item* data yang digunakan pada penelusuran (*search*). Dan yang mendesain aplikasi dapat menggunakan kamus untuk menentukan apakah *item* data sudah disimpan di komputer, dan kalau sudah dengan nama apa *item* data tersebut dapat dipanggil dan aplikasi apa yang digunakan.

Kamus data berguna khusus bagi perlindungan timbulnya kelebihan data. Tanpa kamus data, pemakai dari lain bagian mungkin menyimpan versi identik dari *item* data yang sama pada beberapa lokasi, dimana masing-masing *item* data mempunyai nama yang berbeda.

Operasional komputer dalam organisasi dewasa ini banyak yang sudah menggunakan model kerja jaringan (*network*). Dengan menggunakan data dasar yang sama untuk keperluan informasi masing-masing unit dan manajemen organisasi secara keseluruhan (Drs. Zulkifli Amsyah ; 2005 : 382).

II.10. Teknik Normalisasi

Salah satu topik yang cukup kompleks dalam dunia manajemen *database*adalah proses untuk menormalisasi tabel-tabel dalam *database relasional.* Dengan normalisasi kita ingin mendesain *database relasional* yang terdiri dari tabel-tabel berikut :

1. Berisi data yang diperlukan.
2. Memiliki sesedikit mungkin redundansi.
3. Mengakomodasi banyak nilai untuk tipe data yang diperlukan.
4. Mengefisienkan update.
5. Menghindari kemungkinan kehilangan data secara tidak disengaja/tidak diketahui.

Alasan utama dari normalisasi *database* minimal sampai dengan bentuk normal ketiga adalah menghilangkan kemungkinan adanya "*insertionanomalies"*, "*deletionanomalies",* dan "*updateanomalies".* Tipe-tipe kesalahan tersebut sangat mungkin terjadi pada *database* yang tidak normal.

II.10.1.Bentuk-bentuk Normalisasi

a. Bentuk tidak normal

Bentuk ini merupakan kumpulan data yang akan direkam, tidak ada keharusan mengikuti format tertentu, dapat saja tidak lengkap dan terduplikasi. Data dikumpulkan apa adanya sesuai keadaanya.

b. Bentuk normal tahap pertama (1" Normal Form)

Definisi :

Sebuah table disebut 1NF jika :

- Tidak ada baris yang duplikat dalam tabel tersebut.
- Masing-masing cell bernilai tunggal

Catatan: Permintaan yang menyatakan tidak ada baris yang duplikat dalam sebuah tabel berarti tabel tersebut memiliki sebuah kunci, meskipun kunci tersebut dibuat dari kombinasi lebih dari satu kolom atau bahkan kunci tersebut merupakan kombinasi dari semua kolom.

c. Bentuk normal tahap kedua (2^{nd} normal form)

Bentuk normal kedua (2NF) terpenuhi jika pada sebuah tabel semua atribut yang tidak termasuk dalam primary key memiliki ketergantungan fungsional pada primary key secara utuh.

d. Bentuk normal tahap ketiga (3^{rd} normal form)

Sebuah tabel dikatakan memenuhi bentuk normal ketiga (3NF), jika untuk setiap ketergantungan fungsional dengan notasi X -> A,

dimana A mewakili semua atribut tunggal di dalam tabel yang tidak ada di dalam X, maka :

- X haruslah *superkey* pada tabel tersebut.
- Atau A merupakan bagian dari *primarykey* pada tabel tersebut.

e. Bentuk Normal Tahap Keempat dan Kelima

Penerapan aturan normalisasi sampai bentuk normal ketiga sudah memadai untuk menghasilkan tabel berkualitas baik. Namun demikian, terdapat pula bentuk normal keempat (4NF) dan kelima (5NF). Bentuk Normal keempat berkaitan dengan sifat ketergantungan banyak nilai (*multivalueddependency*) pada suatu tabel yang merupakan pengembangan dari ketergantungan fungsional. Adapun bentuk normal tahap kelima merupakan nama lain dari *Project Join Normal Form*(PJNF).

f. BoyceCode Normal Form (BCNF)

- Memenuhi 1st NF
- Relasi harus bergantung fungsi pada atribut superkey (Kusrini ; 2007 : 39-43).

II.11. UML (*Unified Modeling Language*)

UML singkatan dari *UnifiedModellingLangguage* yang berarti bahasa pemodelan standart. (Chonoles; 2003 : 6) mengatakan sebagai bahasa, berarti *UML* memiliki sintaks dan *semantic*. Ketika kita membuat model menggunakan konsep *UML* ada aturan–aturan yang harus diikuti. Bagaimana elemen pada model-model yang kita buat harus berhubungan satu dengan lainnya harus

mengikuti standart yang ada. *UML* bukan hanya sekedar diagram, tetapi juga menceritakan konteksnya. Ketika pelanggan memesan sesuatu dari sistem, bagaimana transaksinya? Bagaimana sistem mengatasi error yang terjadi? Bagaimana keamanan terhadap sistem yang ada kita buat? Dan sebagainya dapat dijawab dengan *UML*.

UML diaplikasikan untuk maksud tertentu, biasanya antara lain untuk :

1. Merancang perangkat lunak.
2. Sarana komunikasi antara perangkat lunak dengan bisnis.
3. Menjabarkan sistem secara rinci untuk analisa dan mencari apa yang diperlukan sistem.
4. Mendokumentasikan sistem yang ada, proses-proses dan organisasinya.

UML telah diaplikasikan dalam investasi perbankan,lembaga kesehatan, departemen pertahanan, sistem terdistribusi, sistem pendukung alat kerja, retail, sales, dan supplier.

Blok pembangunan utama *UML* adalah diagram. Beberapa diagram ada yang rinci (jenis *timing diagram*) dan lainya ada yang bersifat umum (misalnya diagram kelas). Para pengembang sistem berorentasikan objek menggunakan bahasa model untuk menggambarkan, membangun dan mendokumentasikan sistem yang mereka rancang. *UML* memungkinkan para anggota team untuk bekerja sama dalam mengaplikasikan beragam sistem. Intinya, *UML* merupakan alat komunikasi yang konsisten dalam mensuport para pengembang sistem saat ini. Sebagai perancang sistem mau tidak mau pasti menjumpai *UML*,

baik kita sendiri yang membuat sekedar membaca diagram *UML* buatan orang lain (PrabowoPudjo Widodo Herlawati ; 2011 ; 6).

II.11.1. *Diagram-Diagram UML*

Beberapa literatur menyebutkan bahwa *UML* menyediakan Sembilan jenis diagram, yang lain menyebutkan delapan karena ada beberapa yang digabung, misalnya diagram komunikasi, diagram urutan, dan diagram pewaktuan digabung menjadi diagram interaksi. Namun demikian model-model itu dapat dikelompokan berdasarkan sifatnya yaitu statis atau dinamis. Jenis diagram itu antara lain :

1. Diagram Kelas (*Class Diagram*). Bersifat statis. Diagram ini memperlihatkan himpunan kelas-kelas, antarmuka-antarmuka, kolaborasi, serta relasi-relasi diagram. Diagram ini umu dijumpai pada pemodelan sistem berorentasi objek. Meskipun bersifat statis, sering pula diagram kelas memuat kelas-kelas.
2. Diagram paket (*Package Diagram*) bersifat statis. Diagram ini memperlihatkan kumpulan kelas-kelas merupakan bagian dari diagram komponen.
3. Diagram *UseCase* bersifat statis. Diagram ini memperlihatkan himpunan *use-case* dan aktor-aktor (suatu jenis khusus dari kelas). Diagram ini terutama sangat penting untuk mengorganisasi dan memodelkan perilaku suatu sistem yang dibutuhkan serta diharapkan pengguna.

4. Diagram interaksi dan *Sequence* (urutan). Bersifat dinamis. Diagram urutan adal;ah diagram interaksi yang menekankan pada pengiriman pesan dalam waktu tertentu.
5. Diagram komunikasi (*Communication Diagram*) bersifat dinamis. Diagram sebagai pengganti diagram kolaborasi *UML* yang menekankan organisasi *structural*dari objek-objek yang menerima serta mengirim pesan.
6. Diagram *Statechart* (*Statechart Diagram)* bersifat dinamis. Diagram status memperlihatkan keadaan-keadaan pada sistem, memuat status (*State),* transisi kejadian serta aktifitas. Diagram ini terutama penting untuk memperlihatkan sifat dinamis dari antarmuka (*interface*), kelas, kolaborasi dan terutam penting pada pemodelan sistem-sistem yang reaktif.
7. Diagram aktivitas (*Activity Diagram*) bersifat dinamis. Diagram aktivitas adalah tipe khusus dari diagram status yang memperlihatkan aliran dari suatu sistem. Diagram ini terutama penting dalam pemodelan fungsi-fungsi suatu sistem dan member tekanan pada aliran kendali antar objek.
8. Diagram komponen (*Component Diagram*) bersifat statis. Diagram komponen ini memperlihatkan organisasi serta kebergantungan sistem/perangkat lunak pada komponen-komponen yang telah ada sebelumnya. Diagram ini berhubungan diagram kelas dimana komponen dipetakan kedalam satu atau lebih kelas-kelas. Antarmuka-antarmuka serta kolaborasi-kolaborasi.

9. Diagram *Deployment* (*Deployment Diagram)* bersifat statis. Diagram ini memperlihatkan konfigurasi saat aplikasi dijalankan (*runtime)*. Memuat simpul-simpul berserta komponen-komponen yang ada di dalamnya. Diagram *Deployment* berhubungan erat dengan diagram komponen dimana diagram ini memuat satu atau lebih komponen-komponen. Diagram ini sangat berguna saat aplikasi kita berlaku sebagai aplikasi yang dijalankan pada banyak mesin (*distributed computing).*

Kesembilan diagram ini tidak mutlak harus digunakan dalam pengembangan perangkat lunak, semuanya dibuat sesuai dengan kebutuhan.

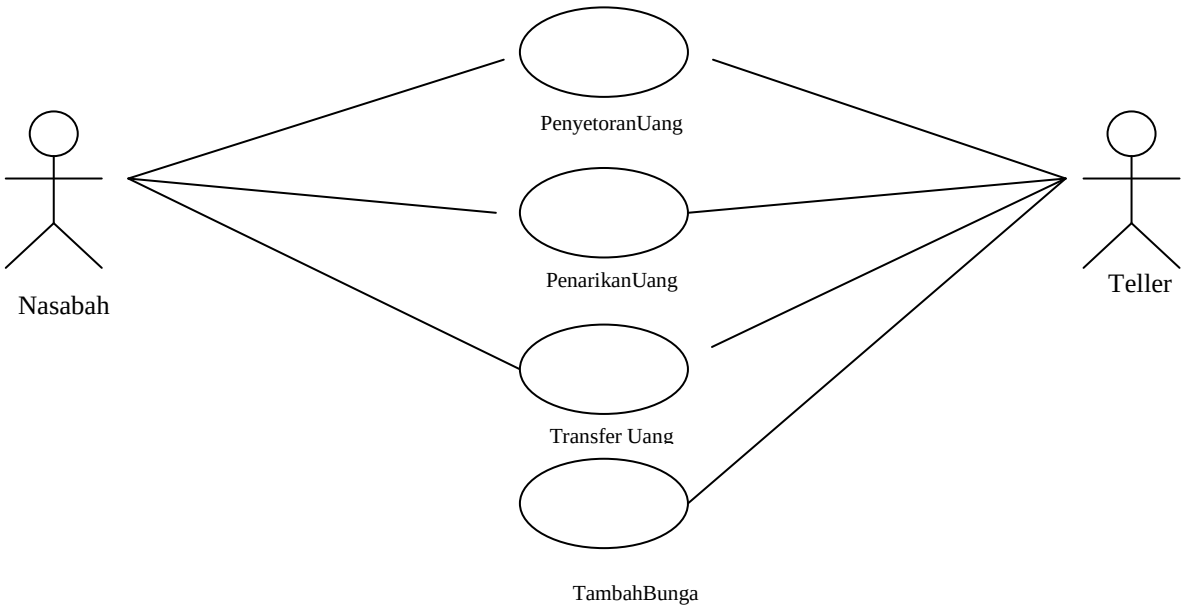
Diagram UseCase (usecase diagram)

*UseCase*menggambarkan *externalview* dari sistem yang akan kita buat modelnya. Menurut Pooley (2005:15) mengatakan bahwa model *usecase* dapat dijabarkan dalam diagram, tetapi yang perlu diingat, diagram tidak identik dengan model karena model lebih luas dari diagram.

komponen pembentuk diagram *usecase*adalah :

a. Aktor (*actor)*, menggambarkan pihak-pihak yang berperan dalam sistem.
b. *UseCase,* aktivitas/ sarana yang disiapkan oleh bisnis/sistem.
c. Hubungan (*Link),* aktor mana saja yang terlibat dalam *usecase* ini.

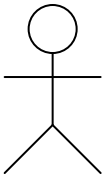
Gambar di bawah ini merupakan salah satu contoh bentuk diagram *usecase.*



Gambar II.2. Diagram *UseCase*
Sumber : ProbowoPudjo Widodo (2011:17)

1. Aktor

Menurut Chonoles (2003 :17) menyarankan sebelum mebuatusecase dan menentukan aktornya, agar mengidentifikasi siapa saja pihak yang terlibat dalam sistem kita. Pihak yang terlibat biasanya dinamakan *stakeholder*.



Gambar II.3. Aktor
Sumber : ProbowoPudjo Widodo (2011:17)

2. *UseCase*

Menurut Pilone (2005 : 21) *usecase* menggambarkan fungsi tertentu dalam suatu sistem berupa komponen kejadian atau kelas. Sedangkan menurut Whitten (2004 : 258) mengartikan *usecase* sebagai urutan langkah-langkah yang secara tindakan saling terkait (skenario) baik terotomatisasi maupun

secara manual, untuk tujuan melengkapi satu tugas bisnis tunggal. *Usecase* digambarkan dalam bentuk *ellips*/oval

Gambar II.4. Simbol *UseCase*
Sumber : ProbowoPudjo Widodo (2011:22)

Usecase sangat menentukan karakteristik sistem yang kita buat, oleh karena itu Chonoles (2003:22-23) menawarkan cara untuk menghasilkan *usecase*yang baik yakni :

a. Pilihlah nama yang baik

Usecase adalah sebuah *behaviour*(prilaku), jadi seharusnya dalam frase kata kerja. Untuk membuat namanya lebih detil tambahkan kata benda mengindikasikan dampak aksinya terhadap suatu kelas objek. Oleh karena itu diagram *usecase* seharusnya berhubungan dengan diagram kelas.

b. Ilustrasikan perilaku dengan lengkap.

Usecase dimulai dari inisiasi oleh aktor primer dan berakhir pada aktor dan menghasilkan tujuan. Jangan membuat *usecase* kecuali anda mengetahui tujuannya. Sebagai contoh memilih tempat tidur (*King Size*, *Queen Size*, atau dobel) saat tamu memesan tidak dapat dijadikan *usecase* karena merupakan bagian dari *usecase* pemesanan kamar dan tidak dapat berdiri sendiri (tidak mungkin tamu memesan kamar tidur jenis king tapi tidak memesan kamar hotel).

c. Identifikasi perilaku dengan lengkap.

Untuk mencapai tujuan dan menghasilkan nilai tertentu dari aktor, *usecase* harus lengkap. Ketika memberi nama pada *usecase*, pilihlah frasa kata kerja yang implikasinya hingga selesai. Misalnya gunakan frasa *reserve a room*(pemesanan kamar) dan jangan *reserving a room*(memesan kamar) karena memesan menggambarkan perilaku yang belum selesai.

d. Menyediakan usecase lawan (*inverse)*

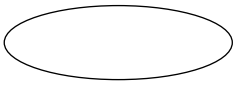
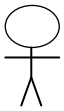
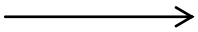
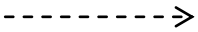
Kita biasanya membutukan *usecase* yang membatalkan tujuan, misalnya pada use*case*pemesanan kamar, dibutuhkan pula *usecase* pembatalan pesanan kamar.

e. Batasi usecase hingga satu perilaku saja.

Kadang kita cenderung membuat *usecase* yang lebih dari satu tujuan aktivitas. Guna menghindari kerancuan, jagalah usecase kita hanya fokus pada satu hal. Misalnya, penggunaan *usecasecheckin* dan *checkout* dalam satu *usecase* menghasilkan ketidakfokusan, karena memiliki dua perilaku yang berbeda.

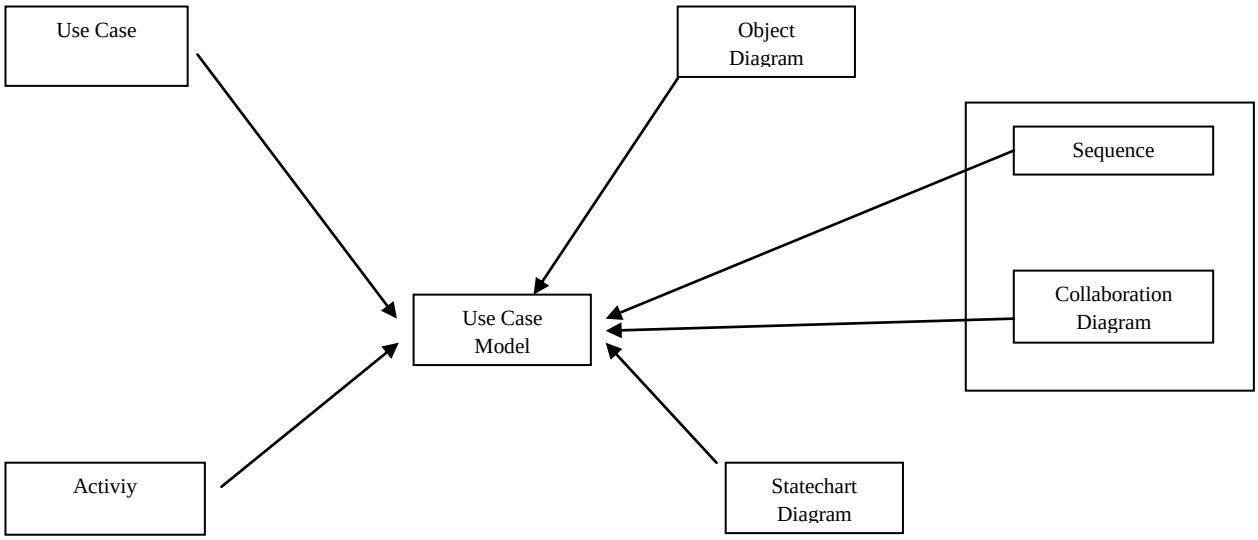
*Usecase*mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan *usecase* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbol-simbol yang digunakan dalam *usecase* diagram, yaitu :

Tabel II.1. Simbol *UseCase*

Gambar	Keterangan
	*Usecase* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukan pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *usecase*.
	Aktor adalah *abstraction*dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktor, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan *usecase*, tetapi tidak memiliki control terhadap *usecase*.
	Asosiasi antara aktor dan *usecase*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengidikasikan aliran data.
	Asosiasi antara aktor dan *usecase* yang menggunakan panah terbuka untuk mengidinkasikan bila aktor berinteraksi secara pasif dengan sistem.
	*Include*, merupakan di dalam *usecase* lain (*required*) atau pemanggilan *usecase* oleh *usecase* lain, contohnya adalah pemanggilan sebuah fungsi program.
	*Extend*, merupakan perluasan dari *usecase* lain jika kondisi atau syarat terpenuhi.

3. Diagram Kelas (*Class Diagram*)

Diagram kelas adalah inti dari proses pemodelan objek. Baik *forwardengineering*maupun *reverse engineering* memanfaatkan diagram ini *forward engineering* adalah proses perubahan model menjadi kode program sedangkan *reverse engineering* sebaliknya merubah kode program menjadi model (ProbowoPudji Widodo; 2011 : 37)



Gambar II.5. Hubungan Diagram Kelas Dengan Diagram *UML*lainya
Sumber : ProbowoPudjo Widodo (2011 : 38)

Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity*atau kardinaliti.

Tabel II.2. *MultiplicityClass Diagram*

Multiplicity	Penjelasan
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimum 4

4. Diagram Aktivitas (*Activity Diagram*)

Diagram aktivitas lebih memfokuskan diri pada eksekusi dan alur sistemdari pada bagaimana sistem dirakit. Diagram ini tidak hanya memodelkan software melainkan memodelkan bisnis juga. Diagram aktivitas menunjukan aktivitas sistem dalam kumpulan aksi-aksi. Ketika digunakan dalam pemodelan *software*, diagram aktivitas merepresentasikan pemanggilan suatu fungsi

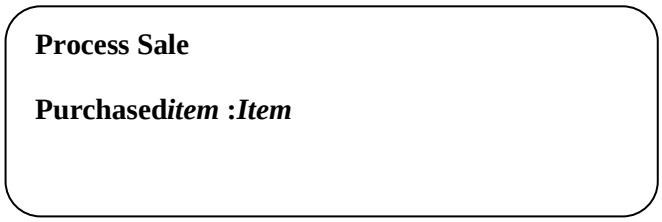
tertentu misalnya *call.* Sedangkan bila digunakan dalam pemodelan bisnis, diagram ini menggambarkan aktivitas yang dipicu oleh kejadian-kejadian diluar seperti pemesanan atau kejadian-kejadian internal misalnya penggajian tiap jumat sore (ProbowoPudji Widodo ;2011 : 143-145).

Aktivitas merupakan kumpulan aksi-aksi. Aksi-aksi nelakukan langka sekali saja tidak boleh dipecah menjadi beberapa langkah-langkah lagi. Contoh aksinya yaitu :

a. Fungsi Matematika
b. Pemanggilan Perilaku
c. Pemrosesan Data

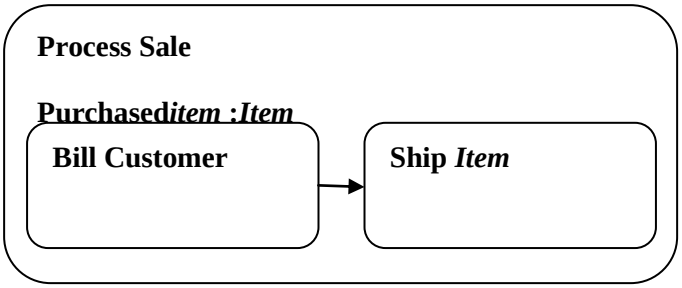
Ketika kita menggunakan diagram aktivitas untuk memodelkan perilaku suatu*classfier*dikatakan kontek dari aktivitas. Aktivitas dapat mengakses atribut dan operasi *classfier*, tiap objek yang terhubung dan parameter-parameter jika aktivitas memiliki hubungan dengan perilaku. Ketika digunakan dengan model proses bisnis, informasi itu biasanya disebut *process-relevant data*. Aktivitas diharapkan dapat digunakan ulang dalam suatu aplikasi, sedangkan aksi biasanya *specific* dan digunakan hanya untuk aktivitas tertentu.

Aktivitas digambarkan dengan persegi panjang tumpul. Namanya ditulis di kiri atas. Parameter yang terlibat dalam aktivitas ditulis dibawahnya.



Gambar II.6. Aktivitas sederhana tanpa rincian
Sumber : ProbowoPudjo Widodo (2011:145)

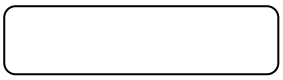
Detail aktivitas dapat dimasukan di dalam kotak. Aksi diperlihatkan dengan symbol yang sama dengan aktivitas dan namanya diletakkan didalam persegi panjang.

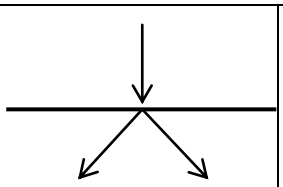
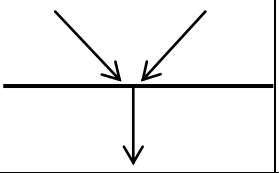
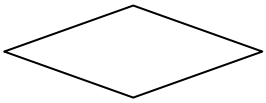
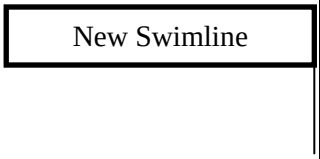


Gambar II.7. Aktivitas dengan detail rincian
Sumber : ProbowoPudjo Widodo (2011:145)

Simbol-simbol yang digunakan dalam *activity diagram*, yaitu :

Tabel II.3. Simbol *Activity Diagram*

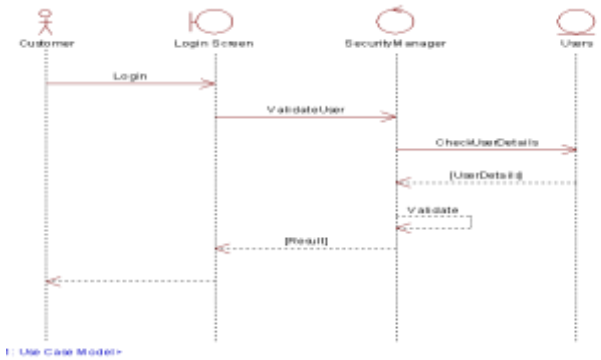
Gambar	Keterangan
	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*Endpoint*, akhir aktifitas.
	*Activites*, menggambarkan suatu proses/kegiatan bisnis.

	*Fork*(Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu.
	*Join* (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
	*DecisionPoints*, menggambarkan pilihan untuk pengambilan keputusan, *true, false.*
	*Swimlane*, pembagian *activity*diagram untuk menunjukkan siapa melakukan apa.

5. *Squence Diagram*

Menurut Douglas (2004 : 174) menyebutkan ada tiga diagram primer UML dalam memodelkan skenario interaksi, yaitu diagram urutan (*sequence diagram),* diagram waktu (*timing diagram)* dan diagram komunikasi (*communication diagram).*

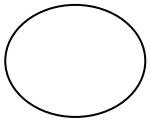
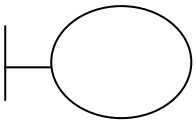
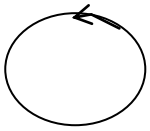
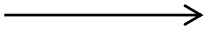
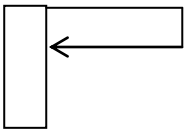
Menurut Pilone (2005 : 174) menyatakan bahwa diagram yang paling banyak dipakai adalah diagram urutan. Gambar II.7. memperlihatkan contoh diagram urutan dengan notasi-notasinya yang akan dijelaskan nantinya.



Gambar II.8. Diagram Urutan
Sumber : ProbowoPudjo Widodo (2011:175)

Simbol-simbol yang digunakan dalam *sequence diagram*, yaitu :

Tabel II.4. Simbol *Sequence Diagram*

Gambar	Keterangan
	*EntityClass*, merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.
	*BoundaryClass*, berisi kumpulan kelas yang menjadi *interface*atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan formentry dan *form*cetak.
	*Control class*, suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek.
	*Message*, simbol mengirim pesan antar *class*.
	*Recursive*, menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.

	*Activation, activation*mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah operasi.
	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline*terdapat *activation*.