

BAB II

TINJAUAN PUSTAKA

II.1. Pengertian Sistem

Mempelajari suatu sistem akan lebih mengena bila mengetahui terlebih dahulu apakah sistem itu. Pengertian tentang sistem pertama kali dapat diperoleh dari defenisi sistem itu sendiri. Jika kita perhatikan dengan seksama, diri kita juga terdiri dari sistem yang berfungsi untuk mengantar kita kepada tujuan hidup kita. Sudah banyak ahli yang mengungkapkan berbagai sistem yang bekerja dalam diri manusia, misalnya sistem kekebalan tubuh untuk menghadapi penyakit cacar dan diptheri. Namun masih banyak pula berbagai sistem yang belum dapat diungkapkan dengan teknologi yang sekarang dimiliki oleh manusia, misalnya sistem kekebalan tubuh untuk menghadapi penyakit AIDS. Contoh sistem lain dalam diri manusia adalah sistem pernapasan, yang berfungsi untuk menyediakan oksigen bagi tubuh dan untuk mengeluarkan zat asam arang yang merupakan sampah hasil pembakaran di dalam tubuh. (Tata Sutabri ; 2012 : 4).

II.2. Pengertian Sistem Informasi

Sistem informasi bukan merupakan hal yang baru, yang baru adalah komputerisasinya. Sebelum ada komputer, teknik penyaluran informasi yang memungkinkan manajer merencanakan serta mengendalikan operasi telah ada. Komputer menambahkan satu atau dua dimensi, seperti kecepatan, ketelitian dan

penyediaan data dengan volume yang lebih besar yang memberikan bahan pertimbangan yang lebih banyak untuk mengambil keputusan.

Sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian yang mendukung fungsi operasi organisasi untuk dapat menyediakan laporan-laporan yang diperlukan oleh pihak luar tertentu (Tata Sutabri ; 2012 : 38).

II.3. Sistem Informasi Geografis

Menurut Kenneth C. Laudan di dalam bukunya Sistem Informasi Manajemen (2008) Sistem Informasi Geografis (*geographic information system-GIS*) adalah kategori khusus dari DSS yang menggunakan teknologi visualisasi data untuk menganalisis dan menampilkan data untuk perencanaan dan pengambilan keputusan dalam bentuk peta digital. Peranti lunak tersebut merakit, menyimpan, memanipulasi, dan menampilkan secara geografis informasi referensi, menghubungkan data dengan titik, garis, dan bidang pada sebuah peta. GIS mempunyai kemampuan membuat model, memungkinkan manajer untuk mengubah data dan secara otomatis memperbarui skenario bisnis untuk mencari solusi yang lebih baik.

GIS membantu pengambilan keputusan yang membutuhkan pengetahuan tentang distribusi penduduk atau sumber daya lain secara geografis. Sebagai contoh, GIS mungkin digunakan untuk membantu pemerintah Negara dan pemerintah lokal menghitung waktu respon bahaya untuk bencana alam, untuk membantu perusahaan eceran mengidentifikasi lokasi pertokoan baru, atau

membantu bank mengidentifikasi tempat terbaik untuk membangun cabang atau memasang terminal ATM baru.

Sesi interaktif manajemen menjelaskan aplikasi GIS untuk mengendalikan tindak kejahatan. CompStat diciptakan oleh Departemen Kepolisian *New York* untuk mengambil data tentang insiden tindak kejahatan dan aktivitas penegakan hukum di setiap sudut kota. CompStat menggunakan peranti lunak GIS untuk menampilkan data mengenai di mana kejahatan berlangsung dan diklaim berhasil mengurangi jumlah rata-rata tindak kejahatan di *New York* dan kota-kota lain.

II.4. Pengertian Quantum GIS

Menurut Prof. Dr. Yupo Chan (2011 : 432) Quantum GIS atau yang sering disingkat menjadi QGIS adalah sebuah aplikasi sistem informasi geografis berbasis *desktop* yang menyediakan fitur untuk menampilkan data, perubahan data dan kemampuan dalam menganalisis data spasial. QGIS dapat berjalan pada sistem operasi Linux, UNIX, Mac OS, dan Windows.

Quantum GIS dapat dibuat dengan bahasa pemrograman C++ dan untuk tampilan grafisnya menggunakan pustaka kode QT-Library. Quantum GIS memungkinkan untuk membentuk integrasi pada Plug-In yang dikembangkan dengan C++ maupun Python.

QT-Library menyediakan tampilan grafis yang dapat berjalan secara *Cross-Platform* dalam *Framework* pengembangan aplikasi yang didukung oleh perangkat lunak lainnya. Quantum GIS memungkinkan untuk dihubungkan atau diintegrasikan dengan berbagai paket perangkat lunak GIS yang bersifat Open-Source lainnya, seperti PostGIS, GRASS, dan MapServer untuk memberikan

fungsionalitas yang ekstensif kepada penggunanya. Quantum GIS secara berkesinambungan terus diperbaiki dan dikembangkan oleh grup pengembang yang aktif dan pengembang sukarela yang secara teratur merilis pembaharuan dan perbaikan pada beberapa kesalahan sistem.

Komponen perangkat lunak GIS dibangun berdasarkan blok-blok sehingga dapat ditambahkan perangkat lunak GIS dan dibentuk dengan baik serta lingkungan pengembangan yang dapat disesuaikan untuk pengguna. Fungsi komponen yang spesifik memberikan dedikasi tugas yang ditambahkan pada lingkungan alat pengembangan GIS, seperti komponen yang memungkinkan untuk memasukkan format data tertentu agar dapat dikonversi, penganalisis data teratur, dan perangkat pemrosesan citra, perangkat pengembangan pengguna di sisi lainnya sebagai fungsi yang spesifik.

II.5. Pengertian PHP

Menurut Andera dan Khairil Nst (2012), **Php** atau *Hypertext Preprocessor* merupakan bahasa berbentuk script yang ditempatkan dalam server dan di eksekusi di dalam server untuk selanjutnya ditrasver dan dibaca oleh client. Php juga bisa disisipkan dalam bahasa HTML. Php pertama kali diciptakan oleh seorang pria berkewarganegaraan Denmark yang bernama Rasmus Lerdorf pada tahun 1995. Banyak programmer yang tertarik untuk mengembangkan php karena bersifat Open Source. Pada awal peluncurannya, Php hanya dibuat untuk diintegrasikan Web Server Apache. Namun sekarang, Php juga dapat bekerja dengan Web Server seperti PWS(Personal Web Server), IIS (Internet Information Server), dan Xitami.

Untuk mengetahui atau sekedar belajar script php, anda tidak perlu menggunakan komputer dengan spek tinggi hanya dengan komputer standar, Anda dapat mempelajari dan mempraktikkan php dalam waktu singkat. Php bersifat Open Source sehingga anda tidak perlu merogoh korek untuk menggunakan perangkat lunak tersebut.

Php sendiri tidak bisa lepas dari database **MySQL**. Oleh karena itu, dalam membuat suatu website dengan bahasa program php, kita membutuhkan **Web Server**.

II.6. Pengertian Database

Database merupakan komponen terpenting dalam pembangunan SI, 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 dieksplorasi secara cepat dan mudah untuk menghasilkan informasi (Budi Sutedjo Dharma Oetomo: 2006: 99).

II.7. Pengertian MySQL

Menurut Wahana Komputer (2010), Mysql pertama kali dirintis oleh seorang programmer *database* bernama Michael Widenius, yang dapat anda hubungi di emailnya monty@analytikerna.

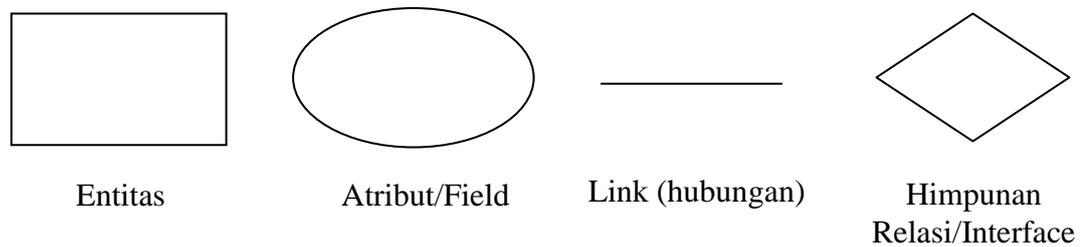
Mysql database server adalah RDBMS (*Relasional Database Management System*) yang dapat menangani data yang bervolume besar. meskipun begitu, tidak menuntut resource yang besar. Mysql adalah *database* yang paling populer diantara *database* yang lain.

MySQL adalah program *database* yang mampu mengirim dan menerima data dengan sangat cepat dan *multiuser*. MySQL memiliki dua bentuk lisensi, yaitu *freeware* dan *shareware*. penulis sendiri dalam menjelaskan buku ini menggunakan *database* ini untuk keperluan pribadi atau usaha tanpa harus membeli atau membayar lisensi, yang berada di bawah lisensi GNU/GPL (*generalpubliclicense*), yang dapat anda *download* pada alamat resminya <http://www.mysql.com>. MySQL sudah cukup lama dikembangkan, beberapa fase penting dalam pengembangan MySQL adalah sebagai berikut :

- MySQL dirilis pertama kali secara internal pada 23 Mei 1995
- Versi windows dirilis pada 8 Januari 1998 untuk windows 95 dan windows NT.
- Versi 3.23 : beta dari Juni 2000, dan dirilis pada January 2001.
- Versi 4.0 : beta dari Agustus 2002, dan dirilis pada Maret 2003 (unions).

II.8. Entity Relationship Diagram (ERD)

Entity Relationship 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).



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 suatu file yang terpisah yang menyimpan informasi seperti :

- Nama setiap item/jenis/kolom data.
- Struktur data untuk tiap item.
- Program yang menggunakan tiap item
- 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 "*insertion anomalies*",

“*deletion anomalies*”, dan “*update anomalies*”. 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 (1st 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 (2nd 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 (3rd normal form)

Sebuah tabel dikatakan memenuhi bentuk normal ketiga (3NF), jika untuk setiap ketergantungan fungsional dengan notasi $X \rightarrow A$, 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 primary key 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 (*multivalued dependency*) pada suatu tabel yang merupakan pengembangan dari ketergantungan fungsional. Adapun bentuk normal tahap kelima merupakan nama lain dari *Project Join Normal Form* (PJNF).

f. Boyce Code 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 *Unified Modelling Language* 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 lainnya ada yang bersifat umum (misalnya diagram kelas). Para pengembang sistem berorientasikan 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 mensupport 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 (Prabowo Pudjo 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 dikelompokkan 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 umum dijumpai pada pemodelan sistem berorientasi objek. Meskipun bersifat statis, sering pula diagram kelas memuat kelas-kelas.
2. Diagram paket (*PackageDiagram*) bersifat statis. Diagram ini memperlihatkan kumpulan kelas-kelas merupakan bagian dari diagram komponen.
3. Diagram *Use Case* 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 adalah 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 terutama 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 (*run time*). Memuat

simpul-simpul beserta 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.

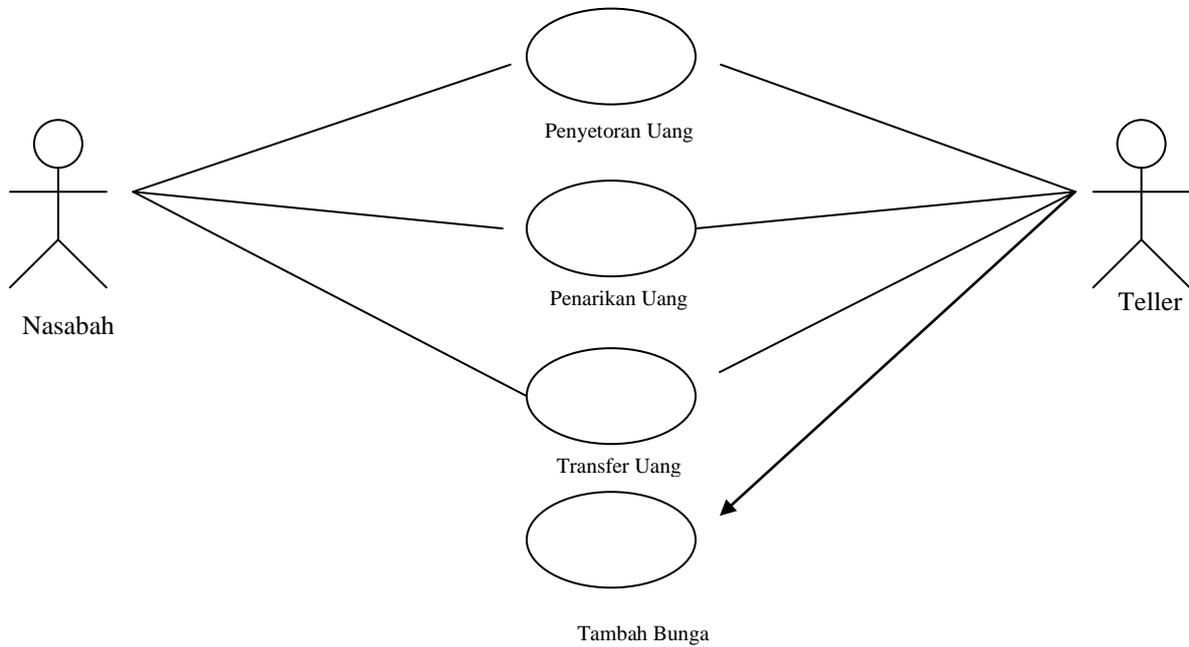
Diagram Use Case (use case diagram)

Use Case menggambarkan *external view* dari sistem yang akan kita buat modelnya. Menurut Pooley (2005:15) mengatakan bahwa model *use case* dapat dijabarkan dalam diagram, tetapi yang perlu diingat, diagram tidak identik dengan model karena model lebih luas dari diagram.

komponen pembentuk diagram *use case* adalah :

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

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

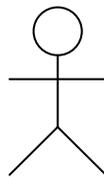


Gambar II.2. Diagram *Use Case*

Sumber : Probowo Pudjo Widodo (2011:17)

1. Aktor

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

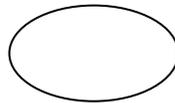


Gambar II.3. Aktor

Sumber : Probowo Pudjo Widodo (2011:17)

2. *Use Case*

Menurut Pitone (2005 : 21) *use case* menggambarkan fungsi tertentu dalam suatu sistem berupa komponen kejadian atau kelas. Sedangkan menurut Whitten (2004 : 258) mengartikan *use case* sebagai urutan langkah-langkah yang secara tindakan saling terkait (skenario) baik terotomatisasi maupun secara manual, untuk tujuan melengkapi satu tugas bisnis tunggal. *Use case* digambarkan dalam bentuk *ellips/oval*



Gambar II.4. Simbol *Use Case*

Sumber : Probowo Pudjo Widodo (2011:22)

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

a. Pilihlah nama yang baik

Use case 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 *use case* seharusnya berhubungan dengan diagram kelas.

b. Ilustrasikan perilaku dengan lengkap.

Use case dimulai dari inisiasi oleh aktor primer dan berakhir pada aktor dan menghasilkan tujuan. Jangan membuat *use case* kecuali anda mengetahui tujuannya. Sebagai contoh memilih tempat tidur (*King Size*, *Queen Size*, atau *dobel*) saat tamu memesan tidak dapat dijadikan *use case* karena merupakan bagian dari *use case* 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 *use case*, 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 use case lawan (*inverse*)

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

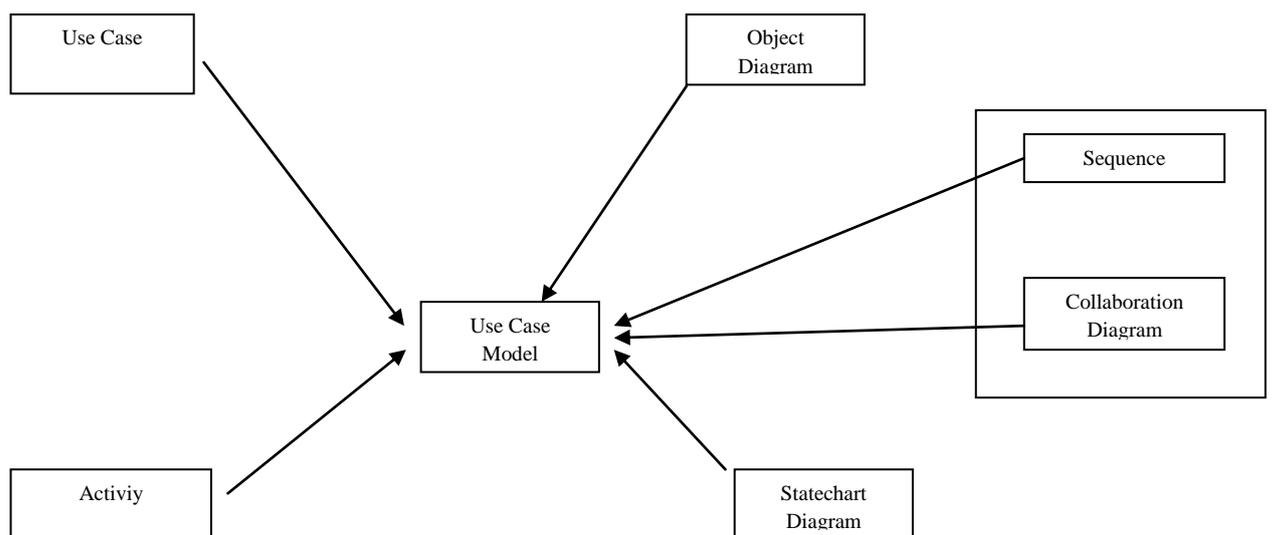
e. Batasi use case hingga satu perilaku saja.

Kadang kita cenderung membuat *use case* yang lebih dari satu tujuan aktivitas. Guna menghindari kerancuan, jagalah *use case* kita hanya fokus pada satu hal. Misalnya, penggunaan *use case* *check in* dan *check out*

dalam satu *use case* menghasilkan ketidakfokusan, karena memiliki dua perilaku yang berbeda.

3. Diagram Kelas (*Class Diagram*)

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



Gambar II.5. Hubungan Diagram Kelas Dengan Diagram *UML* lainnya

Sumber : Probowo Pudjo Widodo (2011 : 38)

4. Diagram Aktivitas (*Activity Diagram*)

Diagram aktivitas lebih memfokuskan diri pada eksekusi dan alur sistem dari pada bagaimana sistem dirakit. Diagram ini tidak hanya memodelkan software melainkan memodelkan bisnis juga. Diagram aktivitas menunjukkan aktivitas

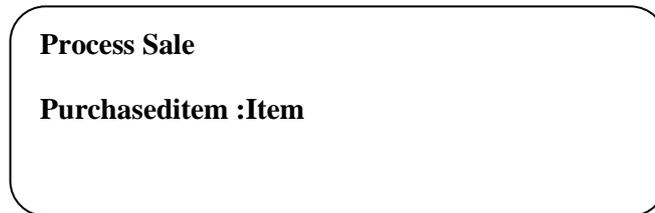
sistem dalam kumpulan aksi-aksi. Ketika digunakan dalam pemodelan *software*, diagram aktivitas merepresentasikan pemanggilan suatu fungsi tertentu misalnya *call*. Sedangkan bila digunakan dalam pemodelan bisnis, diagram ini menggambarkan aktivitas yang dipicu oleh kejadian-kejadian diluar seperti pemesanan atau kejadian-kejadian internal misalnya penggajian tiap jumat sore (Probowo Pudji 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 *classifier* dikatakan kontek dari aktivitas. Aktivitas dapat mengakses atribut dan operasi *classifier*, 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.

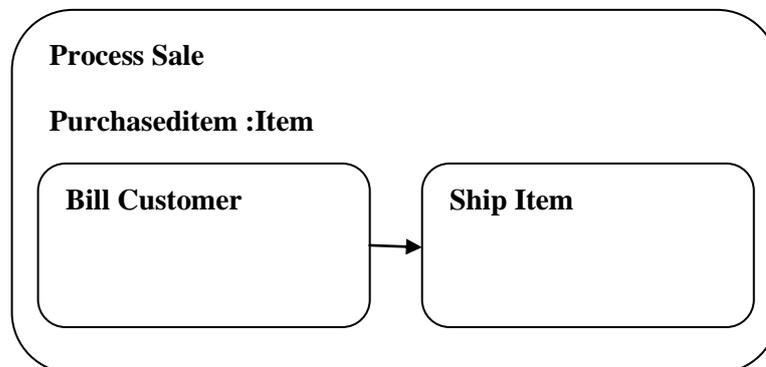
Aktivitas digambarkan dengan persegi panjang tumpul. Namanya ditulis di kiri atas. Parameter yang terlibat dalam aktivitas ditulis dibawahnya.



Gambar II.6. Aktivitas sederhana tanpa rincian

Sumber : Probowo Pudjo Widodo (2011:145)

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



Gambar II.7. Aktivitas dengan detail rincian

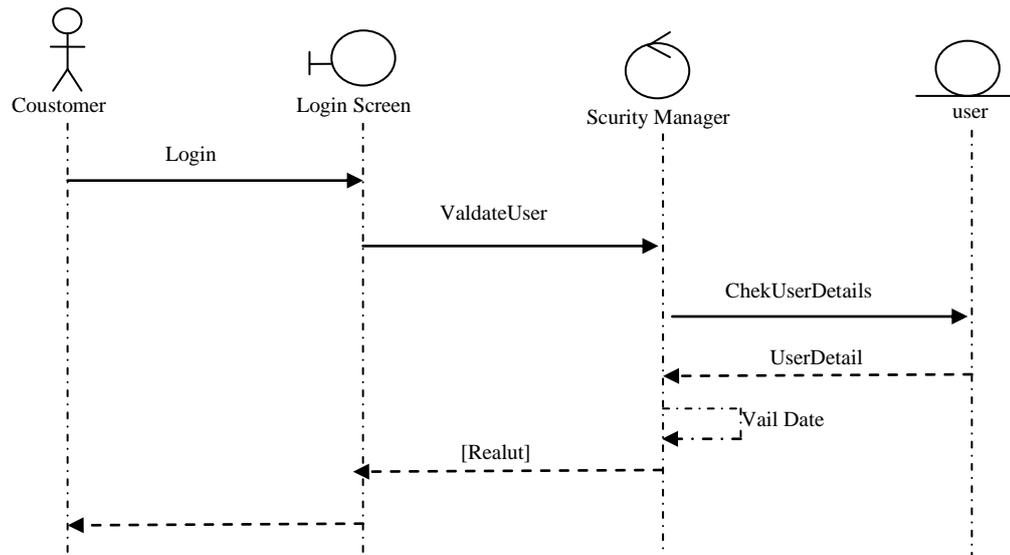
Sumber : Probowo Pudjo Widodo (2011:145)

5. *Sequence Diagram*

Menurut Douglas (2004 : 174) menyebutkan ada tiga diagram primer UML dalam memodelkan scenario 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 : Probowo Pudjo Widodo (2011:175)