BAB II

TINJAUAN PUSTAKA

II.1. Pengertian Sistem

Sistem adalah sebuah tatanan yang terdiri atas sejumlah komponen fungsional (dengan tugas/fungsi khusus) yang saling berhubungan dan secara bersama-sama bertujuan untuk memenuhi suatu proses/pekerjaan tertentu. Sebagai contoh, sistem kendaraan terdiri dari : komponen stater, komponen pengapian, komponen penggerak, komponen pengerem, komponen kelistrikan – *speedometer*, lampu dan lain-lain. Komponen-komponen tersebut diatas memiliki tujuan yang sama, yaitu untuk membuat kendaraan tersebut bisa dikendarai dengan nyaman dan aman (Kusrini ; 2007 : 11).

II.2. Pengertian Informasi

Informasi menjadi penting, karena berdasarkan informasi itu para pengelola dapat mengetahui kondisi obyektif perusahaannya. Informasi tersebut merupakan hasil pengolahan data atau fakta yang dikumpulkan dengan cara tertentu. Informasi disajikan dalam bentuk yang mudah dipahami dan merupakan pengetahuan yang relavan yang dibutuhkan untuk menambah wawasan bagi pemakainya guna mencapai suatu tujuan.

Sebagai contoh, data dapat berupa nama karyawan, jumlah jam kerja, dan lain sebagainya. Jika banyaknya jam kerja dikalikan dengan besarnya upah per jam, maka akan diperoleh gaji kotor. Jika gaji kotor tersebut dijumlahkan, maka

akan diperoleh total gaji kotor. Setelah pemrosesan dilakukan terhadap data, maka akan diperoleh informasi yang dapat mengungkapkan tentang gaji kotor per karyawan dan total biaya gaji yang harus disediakan oleh perusahaan.

Informasi tersebut diperlukan sebagai dasar pertimbangan para pengelola organisasi dalam mengambil keputusan manajerial dan strategis.

Pengolahan data menjadi informasi itu merupakan suatu siklus, yang terdiri dari tahap-tahap sebagai berikut :

1. Pengumpulan data. Pada tahap ini dilakukan suatu proses pengumpulan data yang asli dengan cara tertentu, seperti sampling, data transaksi, data *warehouse*, dan lain sebagainya yang biasanya merupakan proses pencatatan data ke dalam suatu *file*.
2. *Input*. Tahap ini merupakan proses pemasukan data dan prosedur pengolahan data ke dalam komputer melalui alat *input* seperti *keyboard*. Prosedur pengolahan data itu merupakan urutan langkah untuk mengolah data yang ditulis dalam suatu bahasa pemograman yang disebut program.
3. *Pengolahan data*. Tahap ini merupakan tahap di mana data diolah sesuai dengan prosedur yang telah dimasukkan. Kegiatan pengolahan data ini meliputi pengumpulan data, klasifikasi (pengelompokkan), kalkulasi, pengurutan, penggabungan, peringkasan baik dalam bentuk table maupun grafik, penyimpanan dan pembacaan data dari tempat penyimpanan data (Budi Sutedjo Dharma Oetomo ; 2006 : 12).

II.3. 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 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.4. Pengertian Sistem Informasi Manajemen

Sistem informasi manajemen sering disamakan dengan teknologi. Dari definisi di atas dapat disimpulkan bahwa teknologi merupakan bagian dari SIM di samping sumber daya manusia, perusahaan, dan prosedur kerja.

Sistem informasi manajemen adalah suatu sistem yang saling bekerja sama terdiri dari kumpulan orang, alat, serta prosedur dan merupakan satu kesatuan yang saling berinteraksi dan berkesinambungan serta dirancang untuk

mengumpulkan, memilih, menganalisis, mengevaluasi dan mendistribusikan informasi yang baik melalui pembuatan keputusan.

Dalam pendistribusian data maupun informasi dibutuhkan komunikasi yang baik sebagai syarat mutlak untuk melakukan koordinasi kerja, baik secara vertical maupun secara horizontal. Koordinasi merupakan harmonisasi usaha-usaha yang bersifat individual, dan merupakan inti/esensi dari manajemen. Informasi dapat disampaikan secara lisan maupun tulisan misalnya, melalui *meeting*/pertemuan, surat, rapat, bulletin, buku panduan, telepon,*e-mail* dan lain sebagainya (Ida Nuraida ; 2008 : 28).

II.5. Manajemen Personalia

Manajemen personalia adalah ilmu dan seni yang mengatur hubungan, memajukan dan memelihara serta mendukung peranan tenaga kerja agar efektif dan efesien sehingga membantu terwujudnya tujuan perusahaan, karyawan dan masyrakat. Sumber daya manusia, dalam hal ini ikut menentukan tercapainya tujuan suatu organisasi sehingga perlu mendapat perhatian serius. Bagian personalia atau kepegawaian memegang peranan penting, tugasnya adalah mengatur, membina, menggerakkan, mengarahkan, serta mengembangkan pegawai agar mampu menyelesaikan tugas-tugasnya secara efektif dan efesien guna menunjang tercapainya tujuan perusahaan atau organisasi.

Manajemen personalia atau sumber daya manusia memiliki beberapa ruang lingkup kegiatan antara lain :

1. Pelaksanaan seleksi dan penerimaan yang dilanjutkan dengan penempatan personal baru

2. Pelaksanaan mutasi, promosi, dan pemberhentian personal
3. Pemanfaatan sumber tenaga kerja
4. Pemberian kesempatan mengikuti pendidikan dan latihan
5. Pelaksanaan pelimpahan wewenang dan tanggung jawab (Eeng Ahman ; 2007 : 92).

II.6. Pengertian Perekrutan

Perekrutan diartikan sebagai proses penarikan sejumlah calon yang berpotensi untuk diseleksi menjadi pegawai. Proses ini dilakukan dengan mendorong atau merangsang calon yang mempunyai potensi untuk mengajukan lamaran dan berakhir dengan didapatkannya sejumlah calon atau dapat juga dikatakan sebagai upaya pencarian sejumlah calon karyawan yang memenuhhi syarat dalam jumlah tertentu sehingga dari mereka perusahaan dapat menyeleksi orang yang paling tepat untuk mengisi lowongan kerja yang ada. Jadi, sasaran akhir dan keberhasilan suatu proses penarikan diukur dengan didapatkannya calon yang sangat baik, dan ketidakberhasilan perekrutan berarti tidak didapatkannya calon yang paling berpotensi. Beberapa jumlah pelamar yang didapat sehingga dikatakan proses itu berhasil adalah sangat relative, yang jelas lebih baik bila jumlahnya lebih banyak dibandingkan dengan jabatan yang kosong, karena bilamana semakin banyaj berarti dapat dilakukan seleksi yang teliti (Marihot ; 2007 : 95).

II.7. Tenaga Kerja

Tenaga kerja merupakan faktor produksi insani yang secara langsung maupun tidak langsung menjalankan kegiatan produksi. Faktor produksi tenaga kerja juga dikategorikan sebagai menggantikan manusia sebagai pelaksanaan proses produksi, namun keberadaan manusia mutlak diperlukan.

Dalam faktor produksi tenaga kerja ini terkandung unsur fisik, pikiran, serta kemampuan yang dimiliki oleh tenaga kerja. Oleh karena itu, tenaga kerja dapat dikelompokkan berdasarkan kualitas (kemampuan dan keahlian) dan berdasarkan sifat kerjanya.

1. Tenaga kerja menurut kualitas tenaga kerja
    a. Tenaga kerja terdidik, yaitu tenaga kerja yang memerlukan pendidikan tertentu sehingga memiliki keahlian di bidangnya.
    b. Tenaga kerja terampil, yaitu tenaga kerja yang memerlukan kursus atau latihan bidang-bidang keterampilan tertentu sehingga terampil dibidangnya.
    c. Tenaga kerja tidak terdidik dan tidak terlatih, yaitu tenaga kerja yang tidak melalui pendidikan dan pelatihan. Tenaga kerja ini mungkin menjadi tukang sapu jalan, penjaga sekolah atau pekerjaan yang tidak memerlukan pendidikan dan keterampilan.
2. Tenaga kerja menurut sifat kerja
    a. Tenaga kerja rohani, yaitu tenaga kerja yang menggunakan pikiran, rasa dan karsa.

b. Tenaga kerja jasmani, yaitu tenaga kerja yang menggunakan kekuatan fisik dalam kegiatan produksi (N. Gregory Mankiw ; 2007 : 54).

II.8. Pengertian Java

Bahasa pemrograman Java merupakan karya Suns Microsystem Inc. Rilis resmi level *beta* dilakukan pada November 1995. Dua bulan berikutnya Netscape menjadi perusahaan pertama yang memperoleh lisensi bahasa Java dari Sun. Java adalah bahasa pemrograman berorientasi objek yang berukuran kecil, sederhana dan aman, diinterpretasi atau dioptimalisasi secara dinamis, ber-*bytecode*, arsitektur yang netral, mempunyai *garbage-collector, multithreading*, mempunyai mekanisme penanganan pengecualian, berbasis tipe untuk penulisan program mudah diperluas dinamis serta telah diperuntukan sistem tersebar.

Pada awal pembuatannya, Java dinamakan Oak, kemudian nama Oak dinilai kurang menjual sehingga pada Januari 1995 nama Oak diubah menjadi Java. Sebagai bahasa yang bersifat terbuka, Java didukung oleh banyak *programmer* dari seluruh dunia yang memberikan kontribusinya untuk mengembangkan bahasa Java (Hariyanto ; 2011 : 1).

II.9. Pengertian NetBeans

NetBeans merupakan salah satu proyek *open source* yang disponsori oleh *Sun Microsystem*. Proyek ini bediri pada tahun 2000 dan telah menghasilkan 2 produk, yaitu NetBeanss 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.10. Pengertian Database

Database merupakan kumpulan data yang saling berhubungan, hubungan antar data dapat ditunjukkan dengan adanya *field* kunci dari setiap tabel yang beda.

Dalam satu *file* atau tabel terdapat *record-record* yang sejenis, sama besar, sama bentuk, yang merupakan satu kumpulan entitas yang seragam. Satu *record* terdiri dari *field* yang saling berhubungan menunjukkan bahwa *fiel* tersebut satu pengertian yang lengkap dan disimpan dalam satu *record*. Basis data mempunyai beberapa kriteria penting yaitu :

1. Bersifat data oriented dan bukan program oriented.
2. Dapat digunakan oleh beberapa program aplikasi tanpa perlu mengubah basis datanya.
3. Dapat dikembangkan dengan mudah, baik *volume* maupun strukturnya.
4. Dapat memenuhi kebutuhan sistem-sistem baru secara mudah.
5. Dapat digunakan dengan cara-cara yang berbeda.

Prinsip utama *database* adalah pengaturan data dengan tujuan utama fleksibel dan kecepatan pada saat pengambilan data kembali. Adapun ciri-ciri basis data di antaranya adalah sebagai berikut :

1. Efesiensi meliputi kecepatan, ukuran dan ketepatan.
2. Data dalam jumlah besar.
3. Berbagi pakai (dipakai bersama-sama atau *sharebility*).
4. Mengurangi bahkan menghilangkan terjadinya duplikasi dan data yang tidak konsisten (Windu Gata ; 2013 : 19).

II.11. Pengertian MySQL

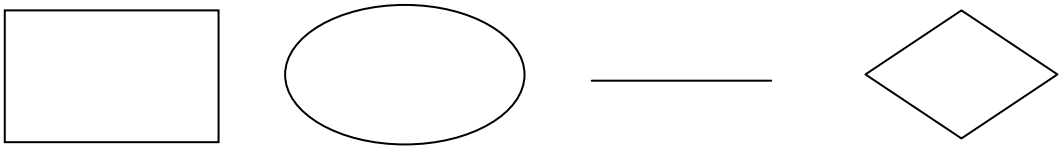
Perangkat lunak MySQL adalah perangkat lunak basis data *server* yang terkenal dan bersifat *open-source* dengan dukungan *driver* yang luas dari berbagai *vendor*. MySQL adalah seakuntansi implementasi dari sistem manajemen basis data relasional (RDBMS) yang didistribusikan secara gratis di bawah lisensi 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 basisdata yang telah ada sebelumnya. SQL (*Structured Query Language*). SQL adalah seakuntansi konsep pengoperasian basisdata, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis.

Kehandalan suatu sistem basisdata (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 basisdata *non*-transaksional. Pada modus operasi *non*-transaksional, MySQL dapat dikatakan unggul dalam hal unjuk kerja dibandingkan perangkat lunak pengelolah basis data kompetitor lainnya. 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, dan sejenisnya. Untuk kebutuhan sistem yang ditujukan untuk bisnis sangat disarankan untuk menggunakan modus basisdata transaksional, hanya saja sebagai konsekuensinya unjuk kerja MySQL pada modus transaksional tidak secepat unjuk kerja pada modus *non*-transaksional (Supardi ; 2007 : 97).

II.12. *Entity Relationship Diagram* (ERD)

Entity Relationship Diagram atau ERD adalah alat pemodelan data utama dan akan membantu mengorganisasi data dalam suatu proyek ke dalam entitas-entitas dan menentukan hubungan antarentitas. Proses memungkinkan analis menghasilkan struktur basisdata yang baik sehingga data dapat disimpan dan diambil secara efisien (Janner Simarmata ; 2010 : 67).



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

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

II.13. Kamus Data

Kamus data (data *dictionary*) mencakup definisi-definisi dari data yang disimpan di dalam basis data dan dikendalikan oleh sistem manajemen basis data. Figur 6.5 menunjukkan hanya satu tabel dalam basis data jadwal. Struktur basis data yang dimuat dalam kamus data adalah kumpulan dari seluruh definisi *field*, definisi tabel, relasi tabel, dan hal-hal lainnya. Nama *field* data, jenis data (seperti teks atau angka atau tanggal), nilai-nilai yang valid untuk data, dan karakteristik-karakteristik lainnya akan disimpan dalam kamus data. Perubahan-perubahan pada struktur data hanya dilakukan satu kali di dalam kamus data, program-program palikasi yang mempergunakan data tidak akan ikut terpengaruh (Raymond McLeod ; 2008 : 171).

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 beberapai item data bersangkutan. Itu adalah fungsi dari kamus data.

Kamus data adalah suatu file yang terpisah yang menyimpan informasi seperti :

1. Nama setiap item/jenis/kolom data.
2. Struktur data untuk tiap item.
3. Program yang menggunakan tiap item
4. Tingkat keamanan untuk setiap item (Drs. Zulkifli Amsyah ; 2005 : 382).

II.14. 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.14.1.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 (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 -> 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.15. UML (*Unified Modeling Language*)

UML singkatan dari *Unified Modelling Langguage* 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 (Prabowo Pudjo Widodo Herlawati ; 2011 ; 6).

II.15.1. *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 *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 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 di mana komponen dipetakan ke dalam 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 berserta komponen-komponen yang ada di dalamnya. Diagram *Deployment* berhubungan erat dengan diagram komponen di mana 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.

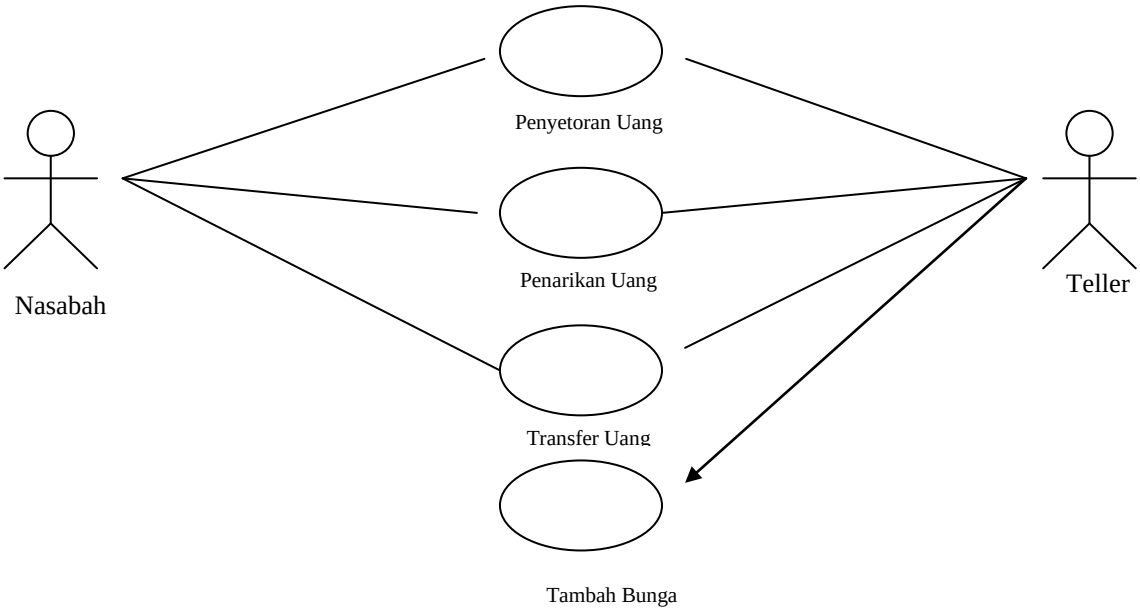
komponen pembentuk diagram *use case* adalah :

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.

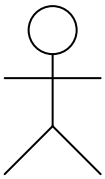
Gambar di bawah ini merupakan salah satu contoh bentuk diagram *use case.*



Gambar II.2. Diagram *Use Case*
Sumber : Probowo Pudjo Widodo (2011:17)

1. Aktor

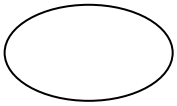
Menurut Chonoles (2003 :17) menyarankan sebelum mebuat use case dan menentukan aktornya, agar mengidentifikasi siapa saja pihak yang terlibat dalam sistem kita. Pihak yang terlibat biasanya dinamakan *stakeholder*.



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

2. *Use Case*

Menurut Pilone (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, *use case* 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)*

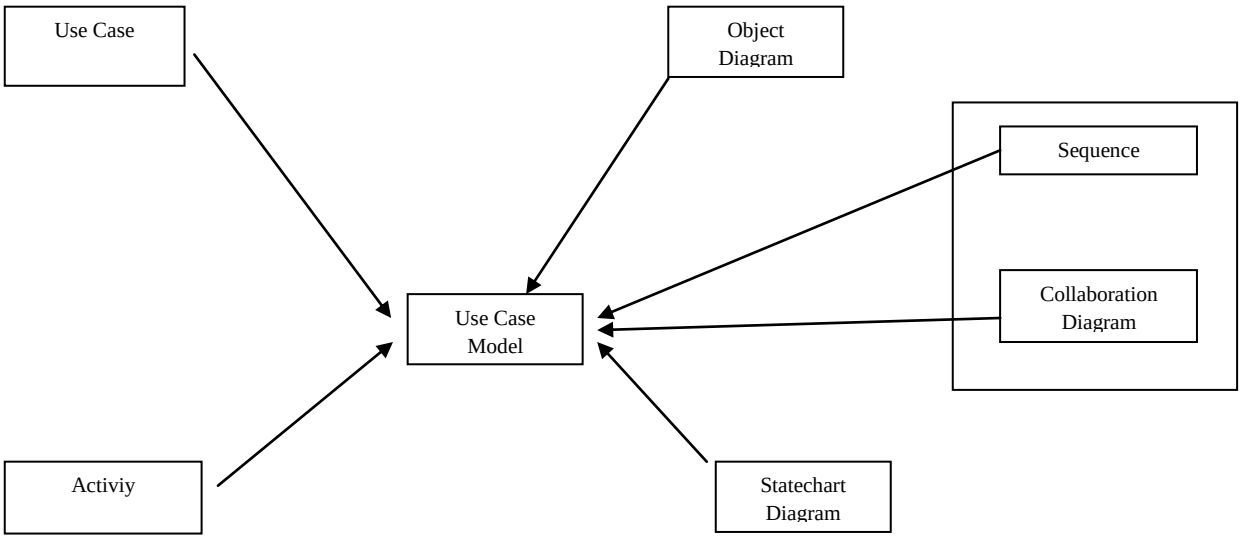
Kita biasanya membutukan *use case* yang membatalkan tujuan, misalnya pada use *case* pemesanan kamar, dibutuhkan pula *use case* pembatalan pesanan kamar.

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* lainya
Sumber : Probowo Pudjo Widodo (2011 : 38)

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 (Probowo Pudji Widodo ;2011 : 143-145).

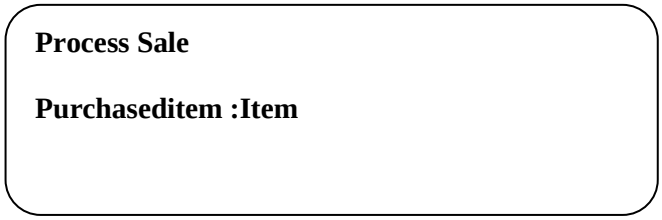
Aktivitas merupakan kumpulan aksi-aksi. Aksi-aksi nelakukan langka sekali saja tidak boleh dipecah menjadi beberapa langkah-langkah lagi. Contoh aksinya yaitu :

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

Ketika kita menggunakan diagram aktivitas untuk memodelkan perilaku suatu *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*.

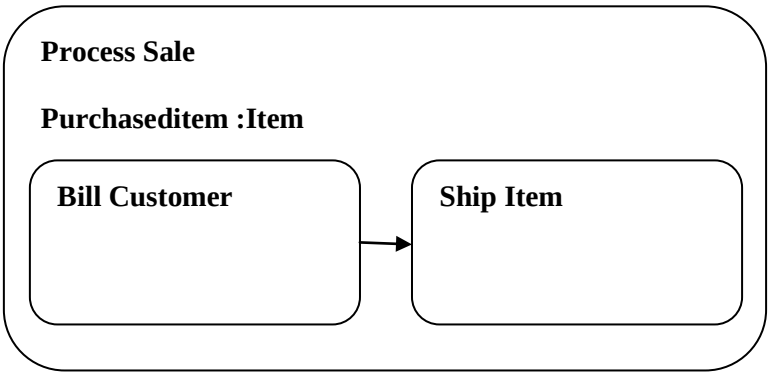
Aktivitas diharapkan dapat digunakan ulang dalam suatu aplikasi, sedangkan aksi biasanya *specific* dan digunakan hanya untuk aktivitas tertentu.

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



Gambar II.6. Aktivitas serderhana tanpa rincian
Sumber : Probowo Pudjo Widodo (2011:145)

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



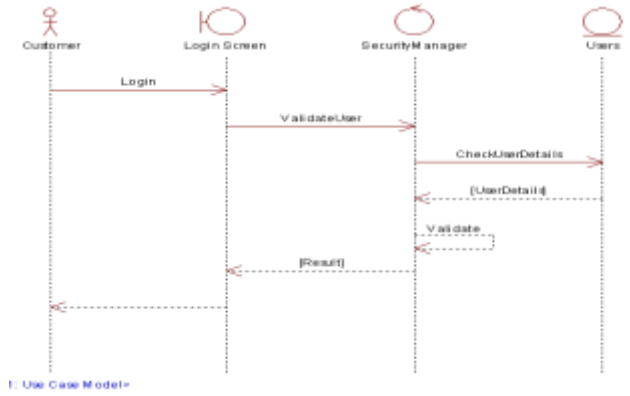
Gambar II.7. Aktivitas dengan detail rincian
Sumber : Probowo Pudjo Widodo (2011:145)

5. *Squence 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.8. memperlihatkan contoh diagram urutan dengan notasi-notasinya yang akan dijelaskan nantinya.



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