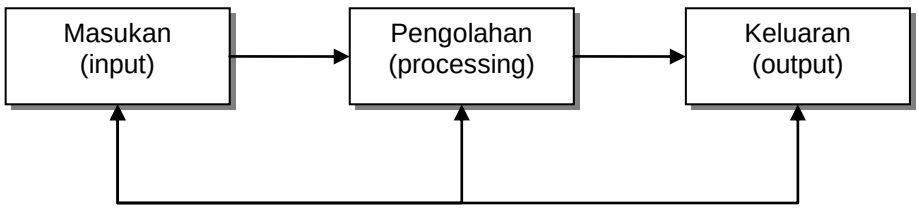
BAB II
TINJAUAN PUSTAKA

II.1. Konsep Dasar Sistem

II.1.1. Pengertian Sistem

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 perangkat elemen yang digabungkan satu sama lainnya untuk suatu tujuan bersama.

Menurut *Scott (1996)*, sistem terdiri dari unsur-unsur seperti masukan (*input*), pengolahan (*processing*), serta keluaran (*output*). Ciri pokok sistem menurut Gapspert ada empat, yaitu sistem itu beroperasi dalam suatu lingkungan, terdiri atas unsur-unsur, ditandai dengan saling berhubungan, dan mempunyai satu fungsi atau tujuan utama.

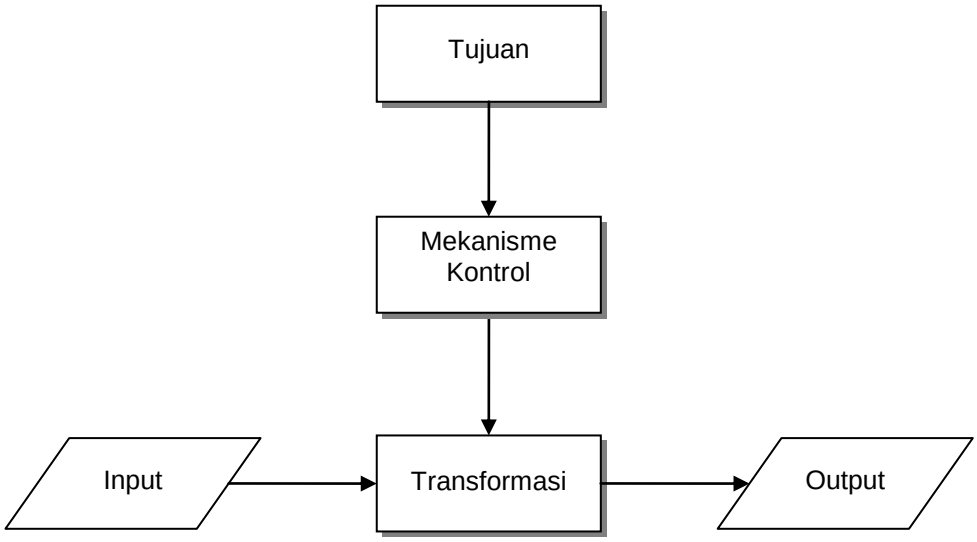


Gambar II.1. Model Sistem

Gambar di atas menunjukkan bahwa sistem atau pendekatan sistem minimal harus mempunyai empat komponen, yakni masukan, pengolahan, keluaran, dan balikan atau *control*.

Sementara *Mc. Leod (1995)* mendefinisikan sistem sebagai sekelompok elemen-elemen yang terintegrasi dengan maksud yang sama untuk mencapai suatu

tujuan. Sumber daya mengalir dari elemen output dan untuk menjamin prosesnya berjalan dengan baik maka dihubungkan dengan mekanisme *control*. Untuk lebih jelasnya elemen sistem tersebut dapat di lihat pada gambar II.2 :



Gambar II.2. Model hubungan elemen-elemen sistem

Banyak ahli mengajukan konsep sistem dengan deskripsi yang berbeda, tetapi pada prinsipnya hampir sama dengan konsep dasar sistem umumnya. *Schronderberg (1971)* dalam *Suradinata (1996)* secara ringkas menjelaskan bahwa sistem adalah :

1. Komponen-komponen yang saling berhubungan satu sama lain.
2. Suatu keseluruhan tanpa memisahkan komponen pembentuknya.
3. Bersama-sama dalam mencapai tujuan.
4. Memiliki input dan output yang dibutuhkan oleh sistem lainnya.
5. Terdapat proses yang mengubah input menjadi output.
6. Menunjukkan adanya entropi.
7. Memiliki aturan.

8. Memiliki subsistem yang lebih kecil.
9. Memiliki deferensi antar subsistem.
10. Memiliki tujuan yang sama meskipun mulainya berbeda. (Hanif Al Fattah, 2007 : 3)

II.1.2. Karakteristik Sistem

Untuk memahami atau mengembangkan suatu sistem, maka perlu membedakan unsur-unsur dari sistem yang membentuknya. Berikut adalah karakteristik sistem yang dapat membedakan suatu sistem dengan sistem yang lainnya :

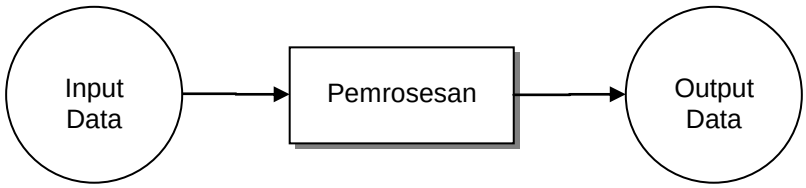
1. Batasan (*boundary*) : Penggambaran dari suatu elemen atau unsur mana yang termasuk di dalam sistem dan mana yang di luar sistem.
2. Lingkungan (*environment*) : Segala sesuatu di luar sistem, lingkungan yang menyediakan asumsi, kendala, dan input terhadap suatu sistem.
3. Masukan (*input*) : Sumber daya (data, bahan baku, peralatan, energi) dari lingkungan yang dikonsumsi dan dimanipulasi oleh suatu sistem.
4. Keluaran (*output*) : sumber daya atau produk (informasi, laporan, dokumen, tampilan layer computer, barang jadi) yang disediakan untuk lingkungan sistem oleh kegiatan dalam suatu sistem.
5. Komponen (*component*) : Kegiatan-kegiatan atau proses dalam suatu sistem yang mentransformasikan input menjadi bentuk setengah jadi (*output*). Komponen ini bisa merupakan subsistem dari sebuah sistem.
6. Penghubung (*interface*) : Tempat di mana komponen atau sistem dan lingkungannya bertemu atau berinteraksi.

7. Penyimpanan (*storage*) : Area yang dikuasai dan digunakan untuk penyimpanan sementara dan tetap dari informasi, energi, bahan baku, dan sebagainya. Penyimpanan merupakan suatu media penyangga di antara komponen tersebut bekerja dengan berbagai tingkatan yang ada dan memungkinkan komponen yang berbeda dari berbagai data yang sama. (Hanif Al Fattah, 2007 : 5)

II.1.3. Pengertian Sistem Inforamasi

Untuk memahami pengertian sistem informasi, harus dilihat keterkaitan antara data dan informasi sebagai entitas penting pembentuk sistem informasi. Data merupakan nilai, keadaan, atau sifat yang berdiri sendiri lepas dari konteks apapun. Sementara informasi adalah data yang telah diolah menjadi sebuah bentuk yang berarti bagi penerimanya dan bermanfaat dalam pengambilan keputusan saat ini atau mendatang (*Davis, 1995*). *Mc Leod (1995)* mengatakan bahwa informasi adalah data yang telah diproses, atau data yang memiliki arti.

Akhirnya Sistem Informasi Manajemen (SIM) dapat didefinisikan sebagai suatu alat untuk menyajikan informasi dengan cara sedemikian rupa sehingga bermanfaat bagi penerimanya (*Kertahadi, 1995*). Tujuannya adalah untuk menyajikan informasi guna pengambilan keputusan pada perencanaan, pemrakarsaan, pengorganisasian, pengendalian kegiatan operasi subsistem suatu perusahaan, dan menyajikan sinergi organisasi pada proses (*Murdick dan Ross, 1993*). Dengan demikian, sistem informasi berdasarkan konsep (*input, processing, output – IPO*) dapat dilihat pada gambar :



Gambar II.3. Konsep sistem informasi (Hanif Al Fattah, 2007 : 9)

II.1.4. Komponen Sistem Informasi

Stair (1992) menjelaskan bahwa sistem informasi berbasis komputer (CBIS) dalam suatu organisasi terdiri dari komponen-komponen berikut :

1. Perangkat keras, yaitu perangkat keras komponen untuk melengkapi kegiatan memasukkan data, memproses data, dan keluaran data.
2. Perangkat lunak, yaitu program dan instruksi yang diberikan ke komputer.
3. *Database*, yaitu kumpulan data dan informasi yang diorganisasikan sedemikian rupa sehingga mudah diakses pengguna sistem informasi.
4. Telekomunikasi, yaitu komunikasi yang menghubungkan antara pengguna sistem dengan sistem komputer secara bersama-sama ke dalam suatu jaringan kerja yang efektif.
5. Manusia, yaitu personel dari sistem informasi, meliputi manajer, analis, programmer, dan operator, serta yang bertanggung jawab terhadap perawatan sistem.
6. Prosedur, yakni tata cara yang meliputi strategi, kebijakan, metode, dan peraturan-peraturan dalam menggunakan sistem informasi berbasis komputer.

Sementara *Burch dan Grudnitski (1986)* berpendapat, sistem informasi yang terdiri dari komponen-komponen di atas disebut dengan istilah blok bangunan (*building block*), yaitu blok masukan (*input block*), blok model (*model*

block), blok keluaran (*output block*), blok teknologi (*technology block*), dan blok kendali (*control block*). Sebagai suatu sistem, keenam blok tersebut masing-masing saling berinteraksi satu dengan yang lainnya membentuk satu kesatuan untuk mencapai sasarannya.

1. Blok Masukan. Input mewakili data yang masuk ke dalam sistem informasi. Input di sini termasuk metode-metode dan media untuk menangkap data yang akan dimasukkan yang dapat berupa dokumen-dokumen dasar.
2. Blok Model. Blok ini terdiri dari kombinasi prosedur, logika, dan model matematika yang akan memanipulasi data input dan data yang tersimpan di basis data dengan cara tertentu untuk menghasilkan keluaran yang diinginkan.
3. Blok Keluaran. Produk dari sistem informasi adalah keluaran yang merupakan informasi yang berkualitas dan dokumentasi yang berguna untuk semua tingkat manajemen serta semua pemakai sistem.
4. Blok Teknologi. Teknologi merupakan kotak alat (*tool box*) dalam sistem informasi. Teknologi digunakan untuk menerima input, menjalankan model, menyimpan dan mengakses data, menghasilkan sekaligus mengirimkan keluaran dan membantu pengendalian dari sistem secara keseluruhan.
5. Blok Database. Database merupakan kumpulan dari data yang saling berhubungan satu dengan yang lainnya, tersimpan di perangkat keras komputer dan digunakan perangkat lunak untuk memanipulasinya.
6. Blok Kendali. Pengendalian perlu dirancang dan diterapkan untuk meyakinkan bahwa hal-hal yang dapat merusak sistem dapat dicegah ataupun bila terlanjur terjadi kesalahan-kesalahan dapat langsung cepat diatasi.

Sementara, menurut pendapat *Davis (1995)*, sistem informasi manajemen terdiri dari elemen-elemen berikut :

1. Perangkat keras komputer (*hardware*)
2. Perangkat lunak (*software*), yang terdiri dari perangkat lunak sistem umum, perangkat lunak terapan, dan program aplikasi.
3. Database
4. Prosedur
5. Petugas operasional. (Hanif Al Fattah, 2007 : 9)

II.2. Franchise dan Laba bersih

II.2.1. Pengertian Franchise

Pengertian franchise adalah duplikasi bisnis yang telah sukses, sehingga bagi mereka yang akan membeli bisnis franchise tidak perlu lagi bersusah payah menjalankan bisnis ini dari awal dan tidak perlu harus jatuh bangun untuk memulai bisnis ini. Mereka hanya menjalankan sistem yang telah berjalan tinggal start up langsung meneruskan bisnis yang memang telah teruji keberhasilannya. (http://www.konsultanwaralaba.com/franchise-keunggulan-berbisnis-franchise/)

II.2.2. Pengertian Laba bersih

Laba bersih *(net income)* adalah selisih lebih pendapatan atas beban-beban dan yang merupakan kenaikan bersih atas modal yang berasal dari kegiatan usaha. (Soemarso S.R, 2009:54)

II.3. Basis Data (*Database*)

Database atau basis data adalah sekumpulan data yang memiliki hubungan secara logika dan diatur dengan seusunan tetentu serta disimpan dalam media

penyimpanan komputer. dat itu sendiri adalah representasi dari semua fakta yang ada pada dunia nyata. database sering digunakan untuk melakukan proses terhadap data-data tesebut untuk menghasilkan informasi tetentu. misalnya dari data nama siswa yang berulang tahun pada hari ini. Tentu saja informasi tersebut akan anda dapatkan dari sofware pemroses database dengan cara anda memberikan perintah dalam bahasa tertentu yaitu SQL(*Structured Query Language*).

Pada era kemajuan teknologi seperti sekarang ini, nilai informasi sangatlah penting, terlebih bagi kemajuan perusahaan. Oleh karena itu penggunaan dan penguasaan database sangat penting. Dalam database ada sebutan-sebutan utnuk satuan data yaitu:

1. Karakter, ini adalah satuan data terkecil. data terdiri atau susunan karakter yang pada akhirnya memawakili data yang memiliki arti dari sebuah fakta.
2. *Field*, adalah kumpulan dari karakter yang mewakili fakta tertentu misalnya seperti nama siswa, tanggal lahir, dan lain-lain. Dalam dunia perancangan database, feld juga disebut atribut. Bila dipandang dari sudut pemrogaman berorientasi obyek maka name dan properti type. Properti name atau nama adalah properti dari field yang berisi field yang mewakili data sejenis yang disimpannya. Sedangkan properti type adalah properti yang mengatur tipe data dari data yang akan ditampungnya. Misalnya nama fieldnya adalah nama siswa maka tipe datanya adalah char, bila nama fieldnya adalah tanggal lahir maka tipe datanya adalah date. Field dilihat seperti kolom.

3. Record, adalah kumpulan dari field. Pada record anda dapat menemukan banyak sekali informasi penting dengan cara mengombinasikan field-field yang ada.
4. Tabel, adalah sekumpulan dari record-record yang memiliki kesamaan entity dalam dunia nyata. Kumpulan dari tabel adalah database, wujud fisik sebuah database dalam komputer adalaha sebuah file yang didalamnya terdapat berbagai tingkatan data yang telah disebutkan di atas.
5. File, adalah bentuk fisik dari penyimpanan data. File database berisi semua data yang telah disusun dan diorganisasikan sedemikian rupa sehingga memudahkan pemberian informasi. (Wahana Komputer, 2010 : 24)

II.3.1. Entity Relationship Diagram

Pada dasarnya ERD(*Entity Relationship Diagram*) adalah sebuah diagram yang secara konseptual memetakan hubungan antar penyimpanan pada diagram DFD di atas. ERD ini digunakan untuk melakukan permodelan terhadap struktur data dan hubungannya. Penggunaannya ERD ini dilakukan untuk mengurangi tingkat kerumitan penyusunan sebuah database yang baik.

Entity dapat berarti sebuah obyek yang dapat dibedakan dengan obyek lainnya. Obyek tersebut dapat memiliki komponen-komponen data (atribut atau field) yang membuatnya dapat dibedakan dari obyek yang lain. Dalam dunia database *entity* memiliki atribut yang menjelaskan karakteristik dari entity tersebut. Ada dua macam atribut yang di kenal deskriptif. Hal ini berarti setiap entity memiliki himpunan yang diperlukan sebuah primary key untuk membedakan anggota-anggota dalam himpunan tersebut.

Atribut dapat memiliki sifat-sifat sebagai berikut:

1. *Atomic*, atomik adalah sifat dari atribut yang menggambarkan bahwa atribut tesebut berisi nilai yang spesifik dan tidak dapat dipecah lagi. Contoh dari sifat atomik adalah field status dari tabel karyawan yang hanya berisi menikah atau single
2. *Multivalued,* sifat ini menandakan atribut ini bisa memiliki lebh dari satu nilai untuk tiap entity tertentu. Misalnya adalah field hobi, hodi dari tiap karyawan mungkin dan hampir pasti lebih dari satu. Misalnya karyawan A mimiliki hobi membaca , nonton TV dan bersepeda.
3. *Composite,* atribut yang bersifat komposit adalah atribut yang nilainya adalah gabungan dari beberapa atribut yang bersifat atomik. Contohnya adalah atribut alamat yang dapat dipecah menjadi atribut atomik berupa alamt, kode pos, no telepon, dan kota. (Wahana Komputer, 2010 : 30)

Ada beberapa derajat relasi tabg daoat terjadi, yaitu :

1. *One to one*, menggambarkan bahwa antara 1 anggota entity A hanya dapat berhubungan dengna 1 anggota entity B. Biasanya derajat relasi ini digambarkan dengna simbol 1-1.
2. *One to many*, menggambarkan bahwa 1 anggota entity A dapat memiliki hubungan dengan lebih dari 1 angota entity B. Biasanya derajat relasi ini digambarkan denga simbol 1-N.
3. *Many to many*, menggambarkan bahwa lebih dari satu anggota A dapat memiliki hubungan dengan lebih dari satu angota entity B. Simbol yang digunakan adalah N-N. (Wahana Komputer, 2010 : 31)

II.3.2. Normalisasi

Setelah melalui tahapan di atas atau ERD, maka hasil pada diagram tersebut mulai direlasasikan pada tabel-tabel database. Untuk itu dperlukan sebuah tahapan yang disebut normalisasi. Normalisasi data adalah proses di mana tabel-tabel pada database dites dalam hal kesalingtergantungan di antara field-field pada sebuah tabel. Misalnya jika pada sebuah tabel terdapat ketergantungan terhadap lebih dari satu field dalam tabel tersebut, maka tabel tersebut harus dipecah menjadi banyak tabel.

Pada prose normaslisasi data, aturan yang dijadikan acuan adalah metode ketergantungan fungsional. Teorinya adalah bahwa tiap kolom dalam sebuah tabel selalu memiliki hubungan yang unik dengan sebuah kolom kunci. Misalnya pada sebuah tabel data_siswa ada field nomor induk data field nama siswa serta field tanggal lahir. Maka ketergantungan fungsionalnya dapat dinyatakan sebagai berikut: nmr_induk -> nm_siswa dan nmr_induk -> tgl_lahir. Artinya nm_siswa memiliki ketergantungan fungsional terhadap nmr_induk. Field nm_siswa isinya juga ditentukan oleh field nmr_induk. Maksud dari semua itu adalah nmr_induk adalah field kunci yang menentukan karena tidak ada nomor induk yang sama pada satu sekolah, jadi field nmr_induk dapat dijadikan patokan untuk mengisi nm_siswa dan field lainnya.

Ada beberapa langkah dalam normalisasi tabel, yaitu:

1. *Decomposition*, dekomposisi adalah proses mengubah bentuk tabel supaya memenuhi syarat tertentu sebagai tabel yang baik. Dekomposisi dapat dikatakan berhasi jika tabel yang dikenal dekomposisi bila digabungkan kembali dapat menjadi tabel awal sebelum di –dekomposisi. Dekomposisi

akan sering dilakukan dalam proses normalisasi untuk memenuhi syarat-syaratnya

2. Bentuk tidak normal, pada bentuk ini semua data yang ada pada tiap entity (diambil atributnya) masih ditampung dalam satu tabel besar. Data yang ada pada tabel ini masih ada yang redundansi dan ada juga yang kosong. Semuanya masih tidak tertata rapi.
3. Normal Form pertama(1^{st} Normal Form), pada tahapan ini tabel di-dekomposisi dari tabel bentuk tidak normal yang kemudian dipisahkan menjadi tabel-tabel kecil yang memiliki kriteria tidak memiliki atribut yang bernilai ganda dan komposit. Semua atribut harus bersifat atomik.
4. Normal Form kedua(2^{nd}Normal Form), pada tahapan ini tabel dianggap memenuhi normal kedua jika pada tabel tersebut semua atribut yang bukan kunci primer bergantung penuh terhadap kunci primer tabel tersebut .
5. Normal Form ketiga(3^{rd}Normal Form), setiap atribut pada tabel selain kunci primer atau kunci utama harus bergantung penuh pada kunci utama. Bentuk normal ketiga biasanya digunakan bila masih ada tabel yang belum efisien. Biasanya penggunaan bentuk normal(normalisasi) hanya sampai pada bentuk ketiga, dan tabel yang dihasilkan telah memiliki kualitas untuk membentuk sebuah database yang dapat diandalkan. Semua tabel diatas juga telah memenuhi bentuk normal tahap ketiga. (Wahana Komputer, 2010 : 32)

II.4. Visual Basic 2010

Visual Basic 2010 merupakan salah satu aplikasi pemrograman visual yang paling banyak digunakan dan cukup populer karena kemudahan penggunaannya.

Visual Basic 2010 merupakan salah satu bagian dari produk pemrograman terbaru yang dekeluarkan oleh Microsoft, yaitu Microsoft Visual Studio 2010. Sebagai produk lingkungan pengembangan terintegrasi atau IDE andalan yang dekeluarkan oleh Microsoft, Visual Studio 2010 menambahkan perbaikan-perbaikan fitur dan fitur baru yang lebih lengkap dibandingkan versi Studio pendahulunya, yaitu Microsoft Visual Studio 2008. (Wahana Komputer, 2010:2)

II.5. Database SQL Server

Microsoft SQL Server 2008 adalah sebuah terobosan baru dari *Microsoft* dalam bidang database. *SQL Server* adalah sebuah *DBMS (Database Management System)* yang dibuat oleh *Microsoft* untuk ikut berkecimpung dalam persaingan dunia pengolahan data menyusul pendahuluanya seperti IBM dan *Oracle*. *SQL Server 2008* dibuat pada saat kemajuan dalam bidang *hardware* sedemikian pesat. Oleh karena itu sudah dapat dipastikan bahwa *SQL Server 2008* membawa beberapa terobosan dalam bidang pengolahan dan penyimpanan data. (Wahana Komputer, 2010:2)

II.6. UML

II.6.1. Pengertian UML

Unified Modeling Language (UML) adalah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

UML merupakan salah satu alat bantu yang sangat handal dalam bidang pengembangan sistem berorientasi objek karena UML menyediakan bahasa pemodelan visual yang memungkinkan pengembangan sistem yang membuat *blue print* atas visinya dalam bentuk yang baku. (Yuni Sugiarti, S.T., M.Kom, 2013:34)

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 Pudji Widodo, Herlawati; 2011 : 6-7).

II.6.2. 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. 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 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 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 (*run time).* 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. Pada *UML* dimungkinkan kita menggunakan diagram-diagram lainya misalnya *Data Flow Diagram, Entity Relationship Diagram* dan sebagainya (Probowo Pudji Widodo, Herlawati; 2011 : 10-12).

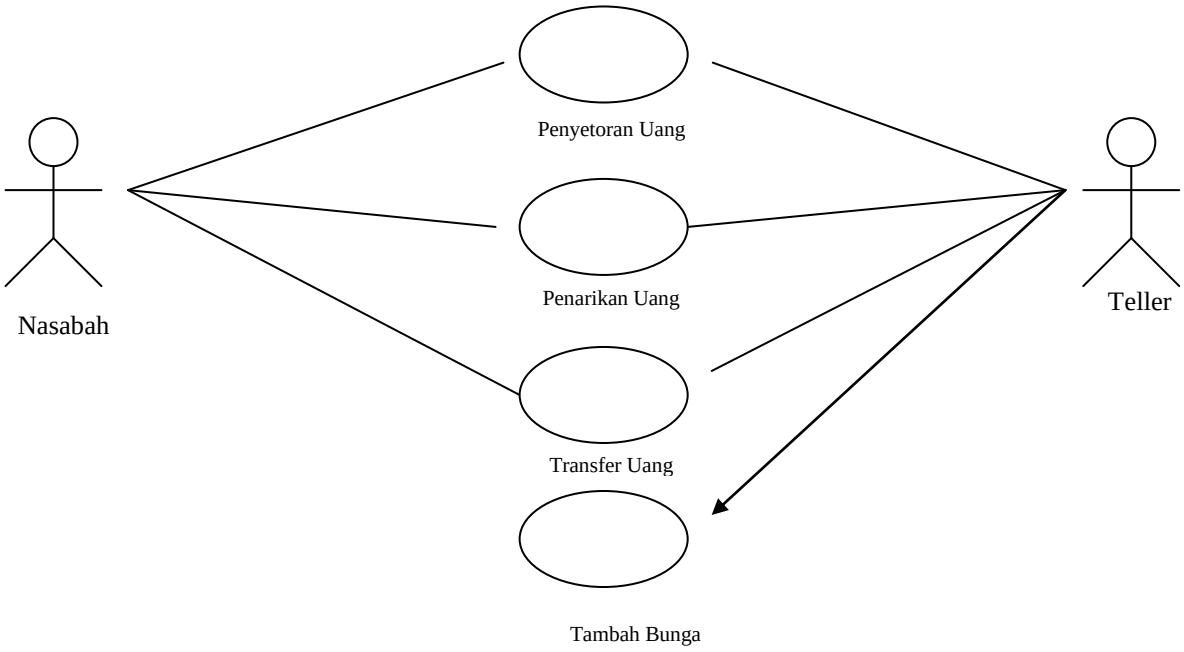
1. *Diagram Use Case (Use Case Diagram)*

   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 indentik 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 II.1. di bawah ini merupakan salah satu contoh bentuk diagram *use case* (Prabowo Pudji Widodo, Herlawati; 2011 : 15-16).

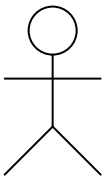


Gambar II.4. Diagram *Use Case*

(Sumber : Prabowo Pudjo Widodo, Herlawati; 2011:17)

2. 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* untuk melihat gambar aktor, lihat pada gambar II.2. sebagai berikut :

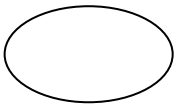


Gambar II.5. Aktor

(Sumber : Prabowo Pudjo Widodo, Herlawati; 2011:17)

3. *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 untuk melihat gambar simbol *use case*, lihat pada gambar II.3. sebagai berikut :



Gambar II.6. Simbol *Use Case*

(Sumber : Prabowo Pudjo Widodo, Herlawati; 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)*

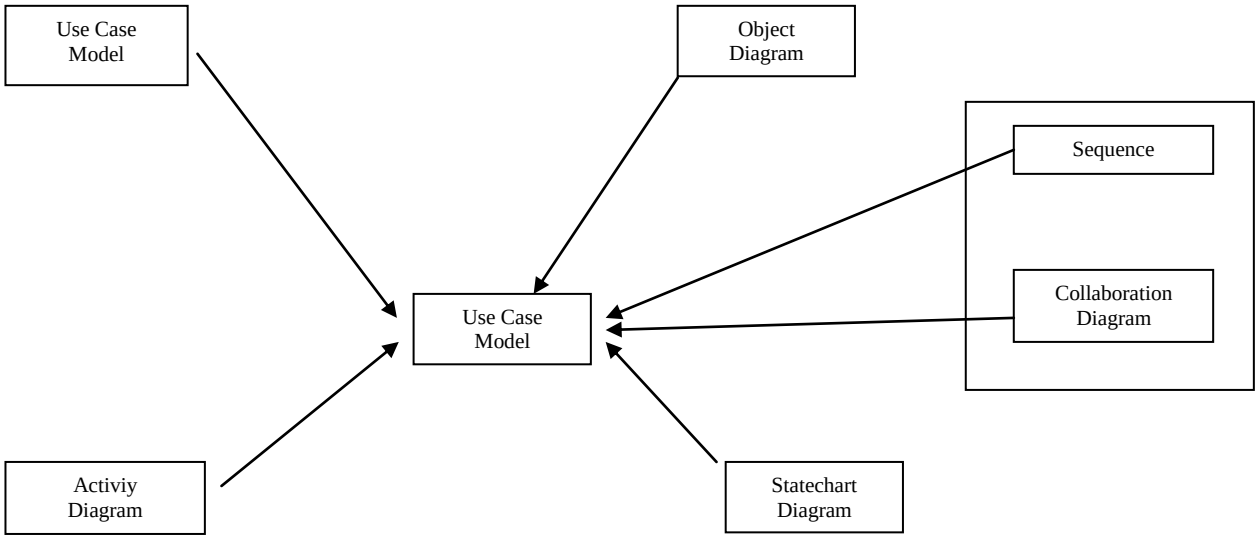
Kita biasanya membutukan *use case* yang membatalkan tujuan, misalnya pada u*se case* pemesanan kamar, dibutuhkan pula *use case* pembatalan pesanan kamar.

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 (Prabowo Pudji Widodo, Herlawati; 2011 : 22-23)

4. 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, Herlawati; 2011 : 37) untuk melihat gambar hubungan diagram kelas dengan diagram *uml* lainya, lihat pada gambar II.4. sebagai berikut :



Gambar II.7. Hubungan Diagram Kelas Dengan Diagram *UML* lainya

(Sumber : Prabowo Pudjo Widodo, Herlawati; 2011 : 38)

5. 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

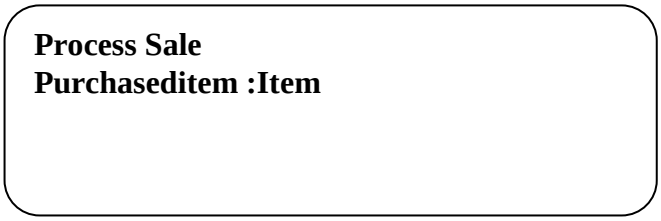
merepresentasikan pemanggilan suatu fungsi tertentu misalnya *call.* Sedangkan bila digunakan dalam pemodelan bisnis, diagram ini menggambarkan aktivitas yang dipicu oleh kejadian-kejadian diluar seperti pemesanan atau kejadian-kejadian internal misalnya penggajian tiap jumat sore (Prabowo Pudjo Widodo, Herlawati ;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.

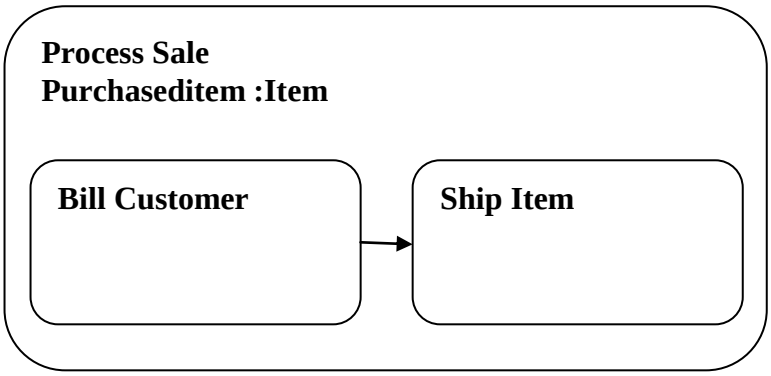
Aktivitas digambarkan dengan persegi panjang tumpul. Namanya ditulis di kiri atas. Parameter yang terlibat dalam aktivitas ditulis dibawahnya. Untuk melihat gambar aktivitas serderhana tanpa rincian, lihat pada gambar II.5. sebagai berikut :



Gambar II.8. Aktivitas Serderhana Tanpa Rincian

(Sumber : Prabowo Pudjo Widodo, Herlawati; 2011:145)

Detail aktivitas dapat dimasukan di dalam kotak. Aksi diperlihatkan dengan symbol yang sama dengan aktivitas dan namanya diletakkan didalam persegi panjang. Untuk melihat gambar aktivitas dengan detail rincian, lihat pada gambar II.6. sebagai berikut :



Gambar II.9. Aktivitas Dengan Detail Rincian

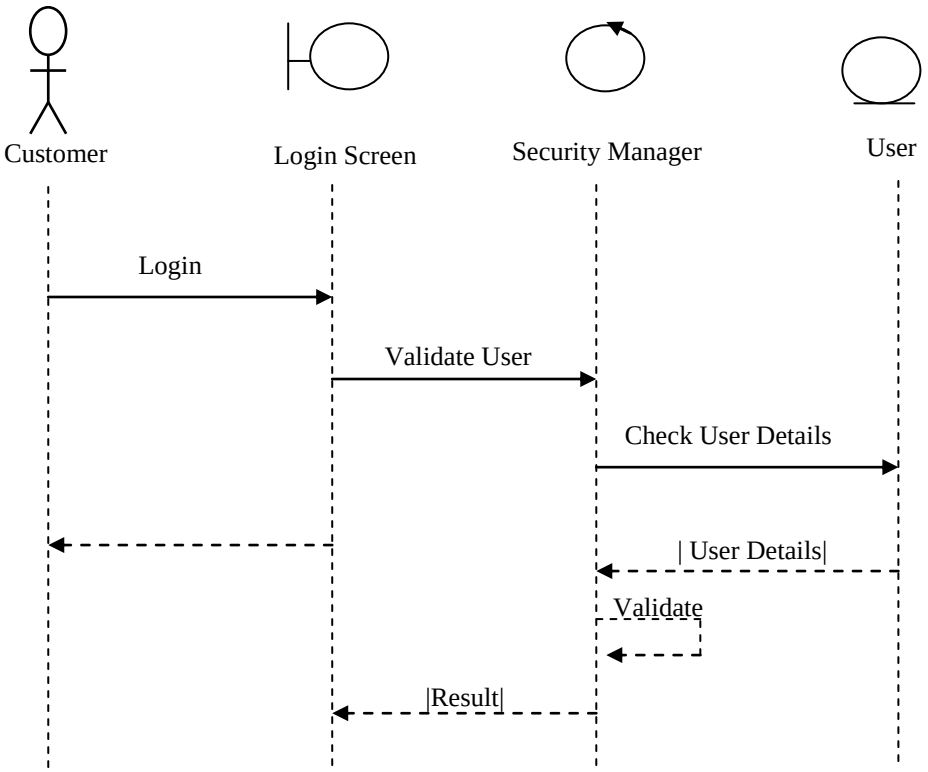
(Sumber : Prabowo Pudjo Widodo, Herlawati; 2011:145)

6. *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 (Prabowo Pudji Widodo, Herlawati; 2011 : 174-175)



Gambar II.10. Diagram Urutan

(Sumber : Prabowo Pudjo Widodo, Herlawati; 2011:175)