BAB II

TINJAUAN PUSTAKA

II.1. Pengertian Sistem

Sistem merupakan kumpulan dari unsur atau elemen-elemen yang saling berkaitan / berinteraksi dan saling mempengaruhi dalam melakukan kegiatan bersama untuk mencapai suatu tujuan tertentu. Contoh : Sistem komputer, terdiri dari software, hardware, dan brainware (Asbon Hendra, S.Kom. ; 2012 : 157).

Menurut H.A. Fatta (2007 : 1), Sistem adalah sekumpulan objek-objek yang saling berelasi dan berinteraksi serta hubungan antar objek bisa dilihat sebagai satu kesatuan yang dirancang untuk mencapai suatu 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 satu sama lain.

Sedangkan menurut Kusrini (2007 : 12), 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 starter, komponen pengapian, komponen penggerak, komponen pengerem, komponen kelistrikan – *speedometer* dan lain-lain (Kusrini, M.Kom ; 2007 : 12).

II.1.1. Karakteristik Sistem

a. Komponen (*Component*)

Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, bekerja sama membentuk suatu kesatuan. Komponen-komponen sistem dapat berupa suatu subsistem atau bagian-bagian dari sistem. Setiap sistem, tidak perduli betapa pun kecilnya, selalu mengandung komponen-komponen atau subsistem-subsistem. Setiap subsistem mempunyai sifat-sifat dari sistem untuk menjalankan suatu fungsi tertentu dan mempengaruhi fungsi sistem secara keseluruhan. Suatu sistem dapat mempunyai suatu sistem yang lebih besar yang disebut supra sistem. Sebagai contoh, suatu perusahaan dapat disebut suatu sistem dan industri merupakan suatu sistem yang lebih besar dapat disebut supra sistem. Kalau dipandang industri sebagai suatu sistem, maka perusahaan dapat disebut sebagai subsistem. Demikian juga apabila perusahaan dipandang sebagai sistem, maka sistem akuntansi adalah subsistemnya.

b. Batas Sistem (*Boundary*)

Batas sistem merupakan daerah yang membatasi antara suatu sistem dengan sistem lainnya atau dengan lingkungan luarnya. Batas sistem ini memungkinkan suatu sistem dipandang sebagai suatu kesatuan, karena dengan batas sistem ini fungsi dan tugas dari subsistem yang satu dengan yang lainnya berbeda tapi tetap saling berinteraksi. Batas suatu sistem menunjukkan ruang lingkup (*scope*) dari sistem tersebut.

c. Lingkungan Luas Sistem (*Environment*)

Environtment merupakan segala sesuatu dari luar batas sistem yang mempengaruhi operasi dari suatu sistem. Lingkunga luar sistem ini dapat bersifat menguntungkan atau merugikan. Lingkungan luar yang menguntungkan harus dipelihara dan dijaga agar tidak hilang pengaruhnya, sedangkan lingkunga luar yang merugikan harus dimusnahkan atau dikendalikan agar tidak mengganggu operasi sistem.

d. Penghubung Sistem (*Interface*)

Merupakan media penghubung antara satu subsistem dengan subsistem yang lainnya untuk membentuk satu kesatuan sehingga sumber-sumber daya mengalir dari subsistem yang satu ke subsistem yang lainnya. Dengan kata lain, *output* dari suatu subsistem akan menjadi *input* dari subsistem lainnya.

e. Masukan Sistem (*Input*)

Merupakan energi yang dimasukkan ke dalam sistem. Masukan dapat berupa masukan perawatan (*maintenance input*) adalah energi yang dimasukkan supaya sistem tersebut dapat beroperasi. Masukan sinyal (*signal input*) adalah energi yang diproses untuk didapatkan keluaran. Sebagai contoh, di dalam sistem komputer, program adalah *maintenance input* yang digunakan untuk mengoperasikan komputernya dan data adalah *signal input* untuk diolah menjadi informasi.

f. Keluaran Sistem (*Output*)

Merupakan hasil dari energi yang diolah sistem, meliputi *output* yang berguna, contohnya informasi yang dikeluarkan oleh komputer. Dan *output* yang tidak berguna dikenal sebagai sisa pembuangan, contohnya panas yang dikeluarkan oleh komputer.

g. Pengolah Sistem (*Process*)

Merupakan bagian yang memproses masukan untuk menjadi keluaran yang diinginkan. Contoh CPU pada komputer, bagian produksi yang mengubah bahan baku menjadi barang jadi, serta bagian akuntansi yang mengolah data transaksi menjadi laporan keuangan.

h. Tujuan Sistem (*Goal*)

Setiap sistem pasti mempunyai tujuan ataupun sasaran yang mempengaruhi input yang dibutuhkan dan output yang dihasilkan. Dengan kata lain, suatu sistem akan dikatakan berhasil kalau pengoperasian sistem itu mengenai sasaran atau tujuannya. Jika sistem tidak mempunyai sasaran, maka operasi sistem tidak akan ada gunanya (Asbon Hendra, S.Kom. ; 2012 ; 158 - 160).

II.2. Pengertian Informasi

Informasi merupakan data yang telah diproses menjadi bentuk yang memiliki arti bagi penerima dan dapat berupa fakta, suatu nilai yang bermanfaat.

Jadi, ada suatu proses transformasi data menjadi suatu informasi = *input* – proses – *output*.

Data merupakan *raw material*, untuk suatu informasi. Perbedaan informasi dan data sangat relatif, tergantung pada nilai gunanya bagi manajemen yang memerlukan. Suatu informasi bagi level manajemen tertentu bisa menjadi data bagi manajemen level diatasnya, atau sebaliknya.

Representasi informasi yaitu pelambangan informasi, misalnya representasi biner.

Kuantitas informasi merupakan suatu ukuran informasi, tergantung representasi. Untuk representasi biner, satuannya *bit*, *byte*, *word* dan lain-lain.

Kualitas informasi adalah bias terhadap error karena kesalahan cara pengukuran dan pengumpulan, kegagalan mengikuti prosedur pemrosesan, kehilangan atau data tidak diproses, kesalahan perekaman atau koreksi data, kesalahan *file* histori / *master*, kesalahan prosedur pemrosesan serta ketidak berfungsian sistem.

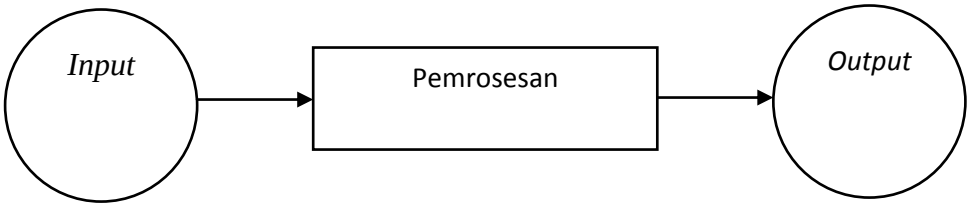
Umur informasi, kapan atau sampai kapan sebuah informasi memiliki nilai / arti bagi penggunanya. Ada *condition information* (mengacu pada titik waktu tertentu) dan *operating information* (menyatakan suatu perubahan pada suatu *range* tertentu).

Nilai informasi, ditentukan dari dua hal, yaitu manfaat dan biaya mendapatkannya. Suatu informasi dikatakan bernilai bila manfaatnya lebih efektif dibandingkan dengan biaya mendapatkannya. Pengukuran nilai informasi

biasanya dihubungkan dengan analisis *cost effetiveness* atau *cost benefit* (Asbon Hendra, S.Kom. ; 2012 ; 167-168).

II.3. Pengertian Sistem Informasi

Suatu sistem terintegrasi yang mampu menyediakan informasi yang bermanfaat bagi penggunanya. Sebuah sistem terintegrasi atau sistem manusia-mesin, untuk menyediakan informasi untuk mendukung operasi, manajemen dalam suatu organisasi. Ada empat operasi dasar dari sistem informasi, yaitu mengumpulkan, mengolah, menyimpan dan menyebarkan informasi. Informasi mungkin dikumpulkan dari lingkungan dalam atau luar dan memungkinkan didistribusikan ke dalam atau luar organisasi. Contoh sebuah sistem informasi penjualan : pengumpulan data transaksi dan faktur penjualan. Sistem ini memanfaatkan perangkat keras dan perangkat lunak komputer, prosedur manual, model manajemen dan basis data (Asbon Hendra, S.Kom ; 2012 ; 168-169).



Gambar II.1. Konsep Sistem Informasi

(Sumber : Hanif Al Fatta ; 2007 ; 9)

II.4 Pengertian SDLC

Menurut Marimin, H. Tanjung dan H. Prabowo (2010 : 61-63), Beberapa ahli sistem menyatakan bahwa SDLC merupakan pengembangan sistem secara

tradisional dan memiliki beberapa tahapan. Department of justice USA (2003) menyatakan bahwa SDLC memiliki 10 tahapan, O'Brien (2004) menyatakan bahwa SDLC memiliki tahapan dan menurut Turban, McLean, dan Weatherbe SDLC memiliki beberapa tahapan. Pada intinya langkah-langkah dalam metodologi SDLC adalah :

1. Mengevaluasi sistem yang ada

   Dengan evaluasi, akan diketahui kekurangan-kekurangan (defisiensi) yang ada didalam sistem. Hal ini dapat dilakukan dengan cara melakukan (wawancara) dengan pengguna yang menggunakan sistem tersebut dan melakukan konsultasi dengan orang-orang yang berkompeten di bidang tersebut.

2. Mendefinisikan kebutuhan sistem baru yang akan dibangun. Kekurangan-kekurangan yang ada pada sistem lama harus dijelaskan secara spesifik sehingga menjadi perhatian untuk perbaikan sistem yang akan dibangun. Selain menganalisis dan mendefinisikan masalah, sistem informasi yang ada juga memprediksi kemungkinan solusi untuk sistem informasi yang akan dibangun atau dikembangkan serta proses organisasinya.

3. Mendesain sistem yang diusulkan

   Rencana-rencana yang akan dilakukan didasarkan pada konstruksi fisik, perangkat keras, dan perangkat lunak yang diperlukan untuk mendukung sistem informasi.

4. Pengembangan sistem baru

Komponen-komponen dan program harus tersedia dan di *install.* Membangun perangkat lunak yang diperlukan untuk mendukung sistem dan melakukan pengujian secara akurat. Melakukan instalasi dan pengujian terhadap perangkat keras dan mengoperasikan perangkat lunak.

5. Penggunaan sistem yang baru

   Hal ini dapat dilakukan dengan berbagai cara. Sistem baru dapat diimplementasikan untuk menggantikan sistem lama. Penerapan sistem baru sebagai pengganti sistem lama dapat dilakukan secara serentak ataupun bertahap.

6. Evaluasi terhadap sistem yang baru

   Evaluasi harus dilakukan terhadap sistem informasi baru yang telah/sedang berjalan. Hal yang dilakukan adalah mengevaluasi sejauh mana sistem yang telah dibangun dan seberapa bagus sistem telah dioperasikan. Pemeliharaan sistem dilakukan dengan sungguh-sungguh dan teliti secara terus-menerus, sehingga sistem informasi yang dibangun dapat bermanfaat bagi organsisi tersebut.

II.5. Pengertian Pemasaran

Perumusan strategi pemasaran didasarkan pada analisis yang menyeluruh terhadap pengaruh faktor-faktor lingkungan eksternal dan internal perusahaan. Lingkungan eksternal perusahaan setiap saat berubah dengan cepat sehingga melahirkan berbagai peluang dan ancaman baik yang datang dari pesaing utama maupun dari iklim bisnis yang senantiasa berubah. Konsekuensi perubahan faktor

eksternal tersebut juga mengakibatkan perubahan faktor internal perusahaan, seperti perubahan terhadap kekuatan maupun kelemahan yang dimiliki perusahaan tersebut.

a. Pengertian Pemasaran

Pemasaran adalah proses kegiatan dipengaruhi oleh berbagai faktor sosial, budaya, politik, ekonomi dan manajerial. Akibat dari pengaruh berbagai faktor tersebut adalah masing-masing individu maupun kelompok mendapatkan kebutuhan dan keinginan dengan menciptakan, menawarkan dan menukarkan produk yang memiliki nilai komoditas (Freddy Rangkuti ; 2006 : 1).

II.6. 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.6.1. 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 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 1^{st} NF
- Relasi harus bergantung fungsi pada atribut superkey (Kusrini, M.Kom ; 2007 : 39-43).

II.7. Pengertian Database

Database atau basis *data* adalah sekumpulan *data* yang memiliki hubungan secara logika dan diatur dengan susunan tertentu serta disimpan dalam

media penyimpanan komputer. Data itu sendiri adalah representasi dari semua fakta yang ada pada dunia nyata. *Database* sering digunakan untuk melakukan proses terhadap data-data tersebut untuk menghasilkan informasi. Dalam *database* ada sebutan-sebutan untuk satuan data yaitu :

1. Karakter, ini adalah satuan data terkecil. *Data* terdiri atas susunan karakter yang pada akhirnya mewakili data yang memiliki arti dari sebuah fakta.
2. *Field*, adalah kumpulan dari karakter yang memiliki fakta tertentu, misalnya seperti nama siswa, tanggal lahir, dan lain-lain.
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 tabel adalah *database* (Wahana Komputer ; 2010 ; 24).

II.7.1. Bahasa *SQL*

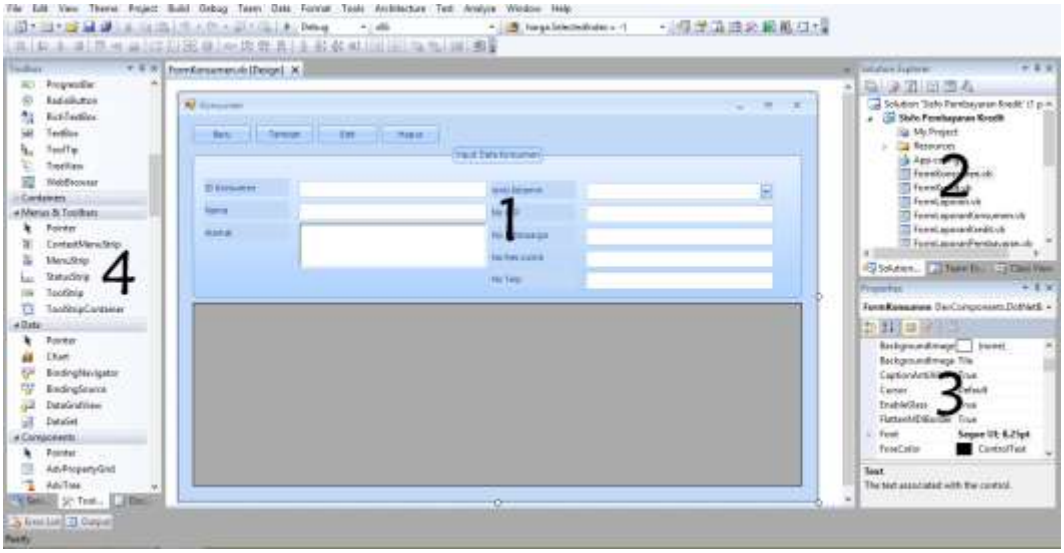
SQL adalah singkatan dari *Structured Query Language*. *SQL* merupakan bahasa terstruktur yang khusus digunakan untuk mengolah *database*. *SQL* pertama kali didefinisikan oleh *American National Standards Institute* (ANSI) pada tahun 1986. Hingga kini *SQL* telah berkembang dan dapat dijalankan pada hampir semua sistem manajemen database termasuk *MySQL*.

Sebagai sebuah bahasa untuk mengolah database, *SQL* memiliki dua komponen utama, yaitu *Data Definition Language (DDL)* dan *Data Manipulation Language (DML).*

DDL digunakan untuk mendefinisikan struktur database serta mengatur bagaimana data dapat diakses. Berikut perintah *DDL* yang umum adalah *CREATE*, *ALTER* dan *DROP*. Perintah *CREATE* digunakan untuk membuat *database* atau tabel. Perintah *ALTER* digunakan untuk mengubah definisi / nama *database* atau tabel. Dan perintah *DROP* digunakan untuk menghapus *database* atau tabel. *DML* digunakan untuk menampilkan data, menambah data, mengubah data, atau menghapus data. Perintah *DML* terdiri atas perintah *SELECT*, *INSERT*, *UPDATE* dan *DELETE*. (A. Ramadhan ; 2006 : 90).

II.8. Sekilas Tentang Visual Basic 2010

Visual basic 2010 adalah bahasa pemrograman klasik, legendaris, dan tiada duanya yang paling banyak dipakai oleh programmer di dunia. Bahasa pemrograman ini dipakai oleh jutaan programmer, dan tercatat sebagai program yang paling banyak dikuasai oleh mayoritas orang. Dari mulai programmer professional yang mencari nafkah dari pembuatan coding dan program, hingga para hobies dan mahasiswa yang membuat program untuk tugas kuliah dan tugas akhir, visual basic memang bisa dihandalkan. Berikut tampilan visual basic 2010 :



Gambar II.2. Tampilan Visual Basic 2010

Sumber : E. Winarno dan A. Zaki (2010:1)

Penjelasan gambar II.2, yaitu :

1. *Form*, merupakan media yang menampung elemen-elemen *form* itu sendiri seperti : *button, combobox, listview, datagridview*, dan lain-lain.
2. *Solution Explorer*, merupakan sebuah bar yang menampilkan file-file yang berada pada project yang sedang dibuat.
3. *Properties*, merupakan bar yang menampilkan properti-properti dari komponen-komponen yang terseleksi.
4. *Toolbox*, merupakan bar yang menyediakan komponen-komponen untuk digunakan dalam membangun aplikasi.

II.9. SQL Server Management Studio

SQL Server Management Studio adalah program yang dibuat oleh *microsoft* untuk membantu *user* ataupun admin melakukan tugas-tugas yang berhubungan dengan *server database*. *SQL Server* memiliki beberapa komponen

penting yang memiliki kegunaannya dalam perancangan *database*, dan melakukan pengaturan sistem secara keseluruhan. Komponen-komponen tersebut adalah :

- *Registered Server*
- *Object Explorer*
- *Query Editor* (Wahana Komputer ; 2010 ; 40).

II.10. *Unified Modeling Language* (UML)

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.10.1. *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. 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 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.

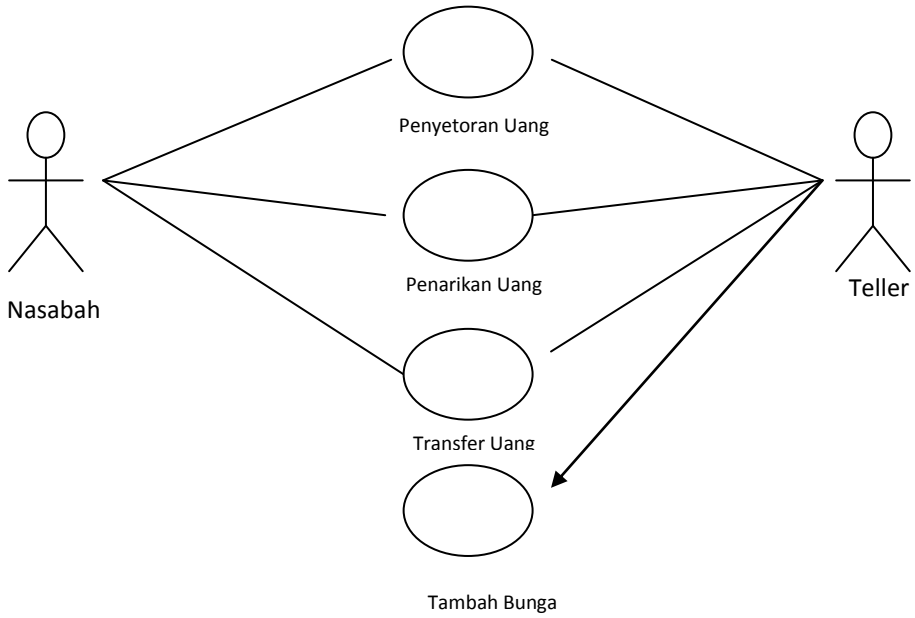
Diagram Use Case (use case diagram)

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.*

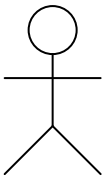


Gambar II.3. Diagram *Use Case*

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*.

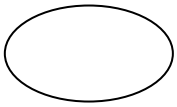


Gambar II.4. Aktor

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.5. 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 u*se case* pemesanan kamar, dibutuhkan pula *use case* pembatalan pesanan kamar.

e. Batasi use case hingga satu perilaku saja.

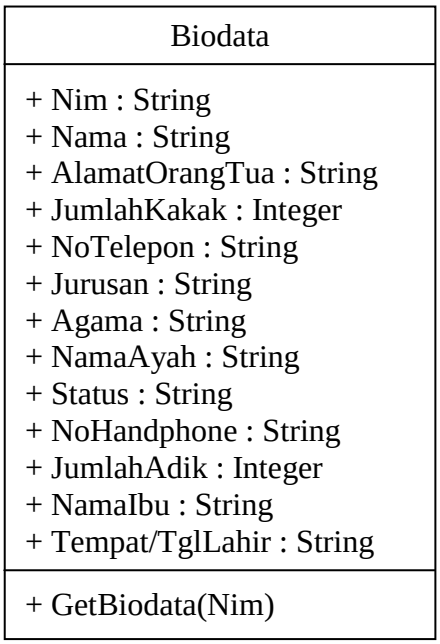
Kadang kita cenderung membuat *use case* yang lebih dari satu tujuan aktivitas. Guna menghindari kerancuan, jagalah use case kita hanya fokus pada satu hal.

3. Diagram Kelas (*Class Diagram*)

Diagram kelas mempunyai dua jenis yaitu *domain class diagram* dan *design class diagram*. Fokus *domain class diagram* adalah pada sesuatu dalam lingkungan kerja pengguna, bukan pada *class* perangkat lunak yang nantinya akan anda rancang. Sedangkan *design class diagram* tujuannya adalah untuk mendokumentasikan dan menggambarkan kelas-kelas dalam pemrograman yang nantinya akan dibangun.

Gambar II.6. Notasi *Domain Diagram Class*

Sumber : E. Triandini dan G. Suardika (2012 : 49 -50)



Gambar II.7. Notasi *Design Diagram Class*

Sumber : E. Triandini dan G. Suardika (2012 : 49 -50)

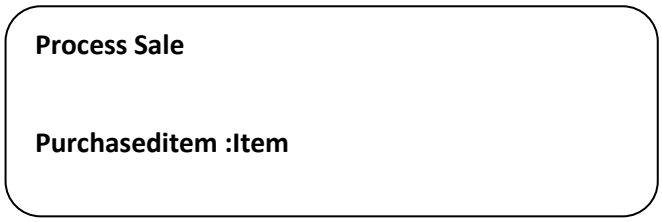
4. Diagram Aktivitas (*Activity Diagram*)

   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

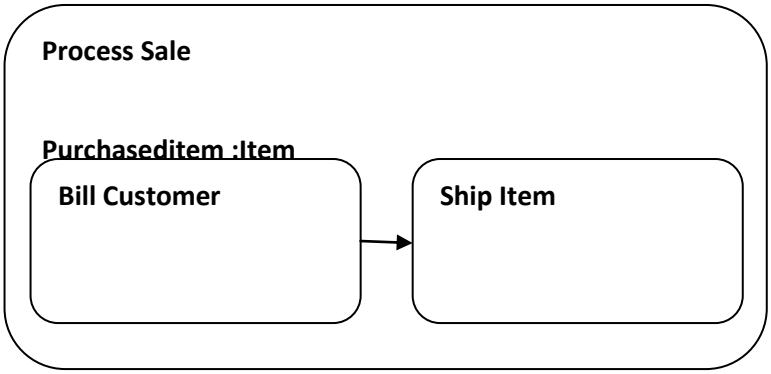
Ketika kita menggunakan diagram aktivitas untuk memodelkan perilaku suatu *classfier* dikatakan kontek dari aktivitas. Aktivitas dapat mengakses atribut dan operasi *classfier*, tiap objek yang terhubung dan parameter-parameter jika aktivitas memiliki hubungan dengan perilaku. Ketika digunakan dengan 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.



Gambar II.8. 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.

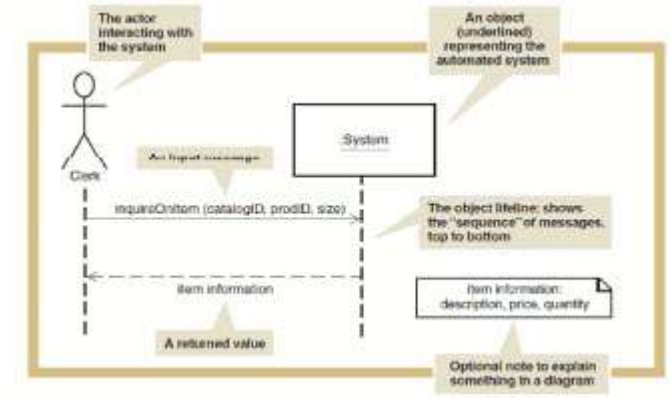


Gambar II.9. Aktivitas dengan detail rincian

Sumber : Probowo Pudjo Widodo (2011:145)

5. *Sequence Diagram*

Menurut John Satzinger, 2010, dalam buku *System Analysis and Design in a Changing World*, "System Sequence Diagram (SSD) adalah diagram yang digunakan untuk mendefinisikan input dan output serta urutan interaksi antara pengguna dan sistem untuk sebuah use case.



.

Gambar II.10. Notasi Sequence Diagram

Sumber : Evi Triandini dan Gede Suardika (2012 : 71)