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

II.1.1. Karakteristik Sistem

1. 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.

2. 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.

3. 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.

4. 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.

5. 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.

6. 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.

7. 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.

8. Tujuan Sistem (*Goal*)

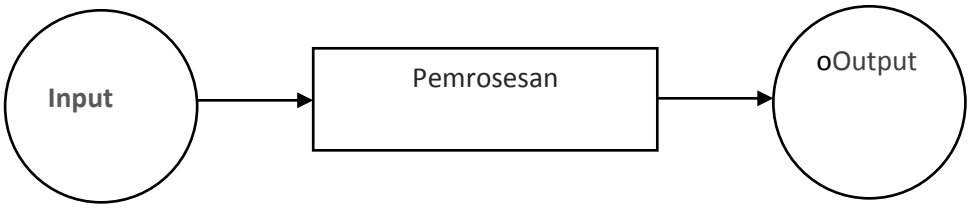
Setiap sistem pasti mempunyai tujuan ataupun sasaran yang mempengaruhi input yang dibutuhkan dan output yang dihasilkan. Dengan kata lain, suatu sistem akan dikatakan berhasil kalau pengoperasian sistem itu mengenai sasaran atau tujuannya. Jika sistem tidak mempunyai sasaran, maka operasi sistem tidak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

II.2. Pengertian Informasi

Untuk memahami makna 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. Sedangkan informasi adalah data yang telah diolah menjadi bentuk yang berarti bagi penerimanya dan bermanfaat dalam pengambilan keputusan saat ini atau mendatang (Hanif Al Fatta ; 2007 ; 6).

II.3. Pengertian Sistem Informasi

Sistem informasi adalah 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 atau perusahaan dan menyajikan sinergi organisasi pada proses dengan demikian sistem informasi berdasarkan konsep (input, processing, output – IPO) dapat dilihat dalam gambar berikut ini :



Gambar II.1 : Konsep Sistem Informasi

(Sumber : *Hanif Al Fatta ; 2007 : 9*)

II.4. Entity Relationship Diagram (ERD)

Entity Relationship Diagram atau ERD adalah gambar atau diagram yang menunjukkan informasi yang dibuat, disimpan, dan digunakan dalam sistem bisnis. Entitas biasanya menggambarkan jenis informasi yang sama. Dalam entitas digunakan untuk menghubungkan antar entitas yang sekaligus menunjukaan hubungan antar data. Pada akhirnya ERD juga bisa digunakan untuk menunjukkan aturan-aturan bisnis yang ada pada sistem informasi yang akan dibangun (*Hanif Al Fatta ; 2007 : 121-122*).

II.5. Kamus Data

Menurut Raymond McLeod Jr dan George P Schell dalam buku Sistem Informasi Manajemen, Kamus data (*Data Dictionary*) mencakup definisi-definisi dari data yang disimpan didalam basis data dan dikendalikan oleh sistem manajemen basis data. Struktur basis data yang dimuat dalam basis 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 aplikasi yang menggunakan data tidak akan terpengaruh.

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

1. 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.

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

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.

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

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

4. 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.

5. 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).

6. 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.8. Visual Basic .Net

Visual basic.net merupakan salah satu bahasa pemrograman yang handal dan banyak digunakan oleh pengembang untuk membangun berbagai macam aplikasi *windows*. *Visual basic* 2008 atau *visual basic* 9 adalah versi terbaru yang

telah diluncurkan oleh microsoft bersama C#, visual C++ dan *visual web developer* dalam satu paket *visual studio* 2008.

Visual basic 2008 merupakan aplikasi pemrograman yang menggunakan teknologi *.Net Framework*. Teknologi *.Net Framework* merupakan komponen *windows* yang terintegrasi serta mendukung pembuatan, penggunaan aplikasi dan halaman *web* (Wahana Komputer. ; 2010 ; 2).

II.9. SQL Server 2008

SQL Server 2008 merupakan DBMS (Database Management System) yang handal dalam mengolah data dengan disertai user interface yang cukup mudah untuk digunakan. Di SQL Server 2008 ini terdapat fitur baru yaitu :

1. *Data Compression*
2. *Change Data Capture*
3. *Filtered Indexes*
4. *Table-Valued Parameter*
5. *Sparse Column*
6. *Data Type* Baru (*Date, Time, Filestream)* (Aryo Nugroho, MCTS dan Smitdev community ; 2008 ; 1).

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.

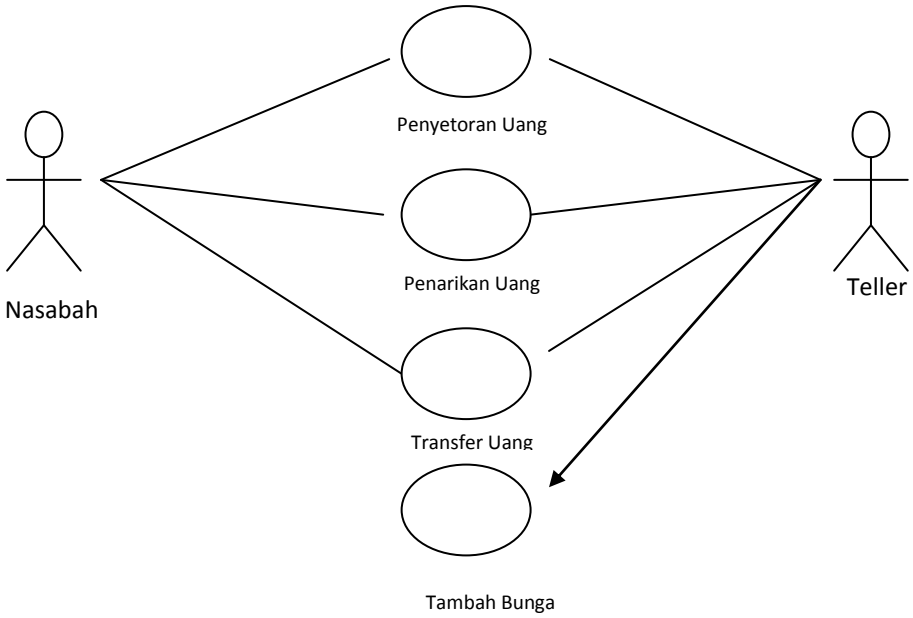
1. *Diagram Use Case (use case diagram)*

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 identik dengan model karena model lebih luas dari diagram.

Komponen pembentuk diagram *use case* adalah :

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

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

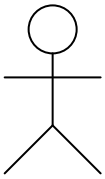


Gambar II.2 : Diagram *Use Case*

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*.

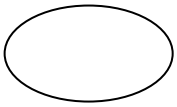


Gambar II.3 : Aktor

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 :

1. 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.

2. 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).

3. 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.

4. 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.

5. 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.

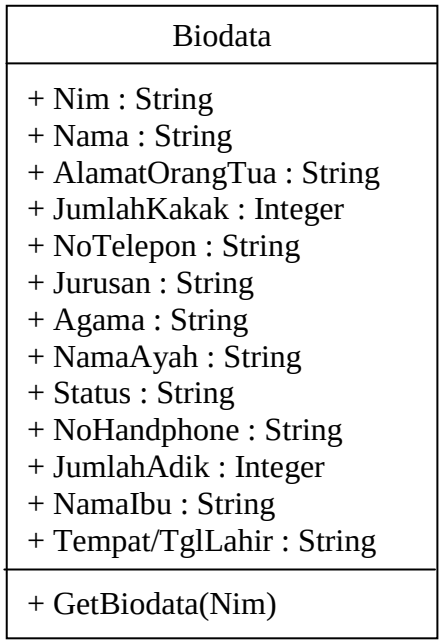
3. Diagram Kelas (*Class Diagram*)

Diagram kelas 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.5 : Notasi *Domain Diagram Class*

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



Gambar II.6 : Notasi *Design Diagram Class*

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

4. Diagram Aktivitas (*Activity Diagram*)

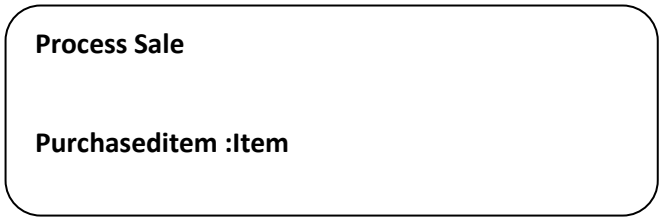
Diagram aktivitas lebih memfokuskan diri pada eksekusi dan alur sistemdari pada bagaimana sistem dirakit. Diagram ini tidak hanya memodelkan software melainkan memodelkan bisnis juga. Diagram aktivitas menunjukan aktivitas sistem dalam kumpulan aksi-aksi. Ketika digunakan dalam pemodelan *software*,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

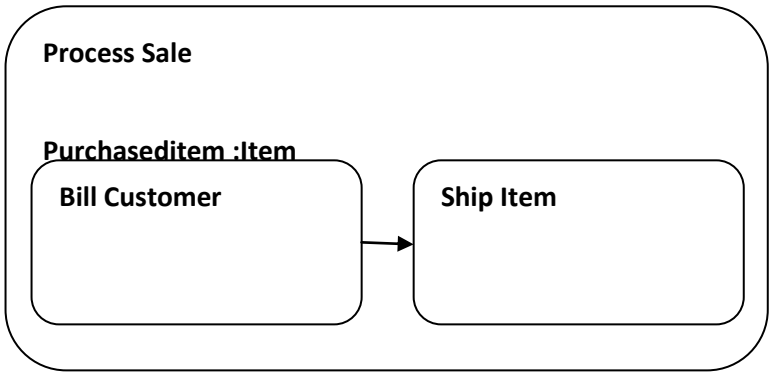
diharapkan dapat digunakan ulang dalam suatu aplikasi, sedangkan aksi biasanya *specific* dan digunakan hanya untuk aktivitas tertentu.



Gambar II.7 : 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.8 : 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 (E. Triandini dan G. Suardika ; 2012 :71).