BAB II

TINJAUAN PUSTAKA

II.1. Pengertian Sistem

Sistem adalah sebuah tatanan yang terdiri atas kumpulan komponen fungsional (dengan tugas / fungsi khusus) yang saling berhubungan dan secara bersama-sama bertujuan untuk mencapai suatu proses / pekerjaan tertentu.

Sebagai contoh, sistem pernafasan terdiri dari : hidung, tenggorokan, paru-paru, pembuluh darah dan darah (Tata Sutabri, S.Kom, MM; 2005 : 6 - 8).

II.1.1. Karakteristik Sistem

1. Komponen (*Component*)

Suatu sistem terdiri dari jumlah komponen yang saling berinteraksi, artinya saling bekerja sama membentuk suatu kesatuan. Komponen-komponen sistem tersebut dapat berupa suatu 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”.

2. Batas Sistem (*Boundary*)

Ruang lingkup sistem merupakan daerah yang membatasi antara sistem dengan sistem yang lain atau sistem dengan lingkungan luarnya. Batasan sistem ini memungkinkan suatu sistem dipandang sebagai satu kesatuan yang tidak dapat dipisah-pisahkan.

3. Lingkungan Luas Sistem (*Environment*)

Bentuk apapun yang ada di luar ruang lingkup atau batasan sistem yang mempengaruhi operasi sistem tersebut disebut lingkungan luar sistem. Lingkungan luar sistem ini dapat bersifat menguntungkan dan dapat juga bersifat merugikan sistem tersebut. Lingkungan luar yang menguntungkan merupakan energi bagi sistem tersebut. Dengan demikian lingkungan luar tersebut harus tetap dijaga dan dipelihara. Lingkungan luar yang merugikan harus dikendalikan. Kalau tidak maka akan mengganggu kelangsungan hidup system tersebut.

4. Penghubung Sistem (*Interface*)

Media yang menghubungkan sistem dengan subsistem lain disebut penghubung sistem atau interface. Penghubung ini memungkinkan sumber-sumber daya mengalir dari satu subsistem ke subsistem yang lain.

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 di olah 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*)

Suatu sistem dapat mempunyai suatu proses yang akan mengubah masukan menjadi keluaran. Contoh, sistem akutansi.

8. Sasaran Sistem (*Objective*)

Suatu system memiliki tujuan atau sasaran yang pasti dan bersifat deterministik. Kalau suatu sistem tidak memiliki sasaran, maka operasi sistem tidak ada gunanya. Suatu sistem dikatakan berhasil apabila mengenai sasaran atau tujuan yang telah direncanakan. (Tata Sutabri, S.Kom, MM; 2005 : 11 - 12).

II.2. Data

Data adalah bahan keterangan tentang kejadian-kejadian nyata atau fakta-fakta yang dirumuskan dalam sekelompok lambing tertentu yang tidak acak, yang menunjukkan jumlah, tindakan atau hal. Data dapat berupa catatan-catatan dalam kertas, buku, atau tersimpan sebagai file dalam basis data. Data menjadi bahan dalam suatu proses pengolahan data. Oleh karena itu, suatu data belum dapat berbicara banyak sebelum diolah lebih lanjut.

Contoh data adalah catatan identitas pegawai, catatan transaksi pembelian, catatan transaksi penjualan, dan lain-lain. (Edhy Sutanta ; 2011 : 13)

II.3. Pengertian Informasi

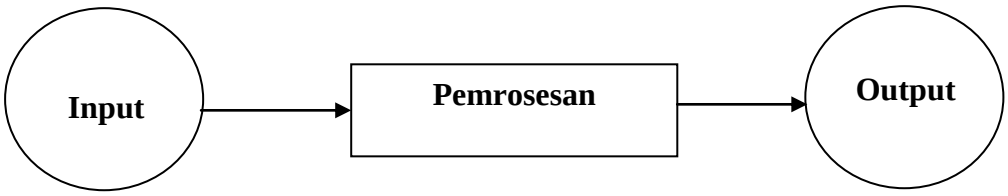
Untuk memahami makna sistem informasi, harus dilihat keterkaitan antara data dan informasi sebagai entitas penting pembentuk sistem informasi. Informasi

merupakan hasil pengolahan data sehingga menjadi bentuk yang penting bagi penerimanya dan mempunyai kegunaan sebagai dasar dalam pengambilan keputusan yang dapat dirasakan akibatnya secara langsung saat itu juga atau secara tidak langsung pada saat mendatang. Untuk memperoleh informasi, diperlukan data yang akan diolah dan unit pengolah.

Contoh informasi adalah daftar pegawai berdasarkan departemen, daftar pegawai berdasarkan golongan, rekapitulasi transaksi pembelian pada akhir bulan, rekapitulasi transaksi penjualan pada akhir bulan, dan lain-lain (Edhy Sutanta ; 2011 : 13 - 14).

II.4. 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.5. Pengolahan Data

Pengolahan data adalah terdiri dari kegiatan-kegiatan penyimpanan data dan penanganan data.

a. Penyimpanan Data (*data storage*)

Penyimpanan data meliputi pekerjaan pengumpulan (*filing*), pencarian (*searching*), dan pemeliharaan (*maintenance*). Data disimpan dalam suatu tempat yang lazim dinamakan "file". File berbentuk map, ordner, disket, tape, hard disk, dan lain sebagainya.

b. Penanganan Data (*data handling*)

Penanganan data meliputi berbagai kegiatan, seperti pemeriksaan (*verifying*), perbandingan (*comparing*), pemilihan (*sorting*), peringkasan (*extracting*), dan penggunaan (*manipulating*). (Tata Sutabri, S.Kom, MM ; 2005 : 21 – 22)

II.6. Stok

Stok atau sering disebut juga dengan persediaan adalah sejumlah bahan-bahan, parts yang disediakan dan bahan-bahan dalam proses yang terdapat dalam perusahaan untuk proses produksi, serta barang-barang jadi/produk yang disediakan untuk memenuhi permintaan dari komponen atau langganan setiap waktu (Assauri : 2008)

II.7. Pupuk

Pupuk adalah suatu bahan yang digunakan untuk mengubah sifat fisik, kimia atau biologi tanah sehingga menjadi lebih baik bagi pertumbuhan tanaman. Dalam pengertian yang khusus pupuk ialah suatu bahan yamg mengandung satu atau lebih hara tanaman. Secara luas pengertian pupuk adalah material yang

ditambahkan pada media tanam atau tanaman untuk mencukupi kebutuhan hara yang diperlukan tanaman sehingga mampu berproduksi dengan baik.

II.8. Entity Relationship Diagram (ERD)

Entity Relationship Diagram atau ERD adalah suatu model data yang berguna untuk memodelkan sistem yang nantinya basis datanya akan dikembangkan. Model ini juga membantu perancang/analisis sistem pada saat melakukan analisis dan perancangan basis data karena model ini dapat menunjukkan macam data yang dibutuhkan dan kerelasian antar data di dalamnya. Sebuah diagram ERD tersusun atas tiga komponen, yaitu :

- Entitas (*Entity*) merupakan objek dasar yang terlibat dalam system.
- Atribut (*Attribute*) berperan sebagai penjelas entitas.
- Kerelasian (*Relationship*) menunjukkan hubungan yang terjadi di antara dua entitas (Edhy Sutanta ; 2011 : 91 – 92).

II.9. Kamus Data

Menurut Raymond McLeod Jr dan George P Schell dalam buku Sistem Informasi Manajemen, Kamus data (*Data Dictionary*) mencakup defenisi-defenisi 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 defenisi *field,* defenisi 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 didalam kamus data, program-program aplikasi yang menggunakan data tidak akan terpengaruh.

II.10. 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 daya yaitu :

1. Karakter, ini adalah satuan data terkecil. *Data* terdiri dari 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 mengkombinasikan *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.11. Sekilas Tentang Java

Java dikenal sebagai bahasa pemrogaman tingkat tinggi yang berorientasi objek, atau lazim disebut dengan istilah *Object Oriented Programming* (OOP)

yang dapat berjalan pada platform yang berbeda, baik di windows, *Linux*, serta sistem operasi lainnya. Jadi kita dapat membuat sebuah aplikasi dengan *java* pada sistem operasi *linux* dan selan jutnyadapat menjalankan atau menginstal aplikasi tersebut pada sistem operasi *windows* dan juga sebaliknya tanpa masalah. Dengan menggunakan *java* kita dapat mengembangkan banyak aplikasi yang dapat digunakan pada lingkugan yang berbeda, seperti pada : *Dekstop, Mobile, Internet,* dan lain-lain.

Berikut ini uraian singkat mengenai paket aplikasi java yang tersedia :

1. J2ME (*Java 2 Micro Edition*)

   Paket instalasi ini dapat digunakan untuk mengembangkan software yang berjalan pada perangkat yang memiliki memori dan sumber daya yang kecil, seperti pada *Handphone,* PDA dan Smartcard.

2. J2SE (*Java 2 Standard Edition*)

   Paket instalasi ini digunakan untuk mengembangkan aplikasi yang berjalan pada lingkungan work station, seperti aplikasi *desktop*.

3. J2EE (*Java 2 Enterprise Edition*)

   Paket instalasi ini dapat digunakan untuk mengembangkan aplikasi pada lingkungan *internet* maupun aplikasi skala *enterprise* (Ridwan Sanjaya, SE, S.Kom ; 2005 : 1-3)

II.12. Sekilas Tentang MySQL

MySQL atau dibaca "*My Sekuel*" adalah suatu RDBMS (*Relational Database Management System*) yaitu aplikasi sistem yang menjalankan fungsi pengolahan data. MySQL pertama kali dikembangkan oleh MySQL AB yang

kemudian diakuisi Sun Microsystem dan terakhir dikelola oleh *Oracle Coorporation.* MySQL dipilih sebagai alternatif database karena ketersediaannya yang mudah di internet, di release untuk berbagai sistem operasi (termasuk Windows dan Linux), dan telah tersedia *connector* yang menghubungkan antara aplikasi *java* dengan database MySQL

II.13. *Unified Modeling Language* (UML)

UML singkatan dari unified modeling language yang berarti bahasa permodelan standard. 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 standard 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 antar 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 ada yang bersifat umum (misalnya diagram kelas). Para pengembang sistem berorientasikan objek menggunakan bahasa model untuk menggambarkan, membangun dan mendokumentasikan sistem yang mereka rancang. UML memungkinkan para anggota team untuk bekerja sama dalam mengaplikasikan beragam sistem. Intinya, UML merupakan alat komunikasi yang konsisten dalam mendukung para pengembang sistem saat ini. Sebagai perancang sistem mau tidak mau pasti menjumpai UML, baik kita sendiri yang membuat ataupun sekedar membaca diagram UML buatan orang lain (Prabowo Pudjo Widodo Herlawati ; 2011 : 6).

II.13.1. 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 perwaktuan digabung menjadi diagram interaksi. Namun demikian model-model itu dapat dikelompokkan berdasarkan sifatnya yaitu statis atau dinamis. Jenis diagram itu antara lain :

1. Diagram kelas bersifat statis. Diagram ini memperlihatkan himpunan kelas-kelas, antarmuka-antarmuka, kolaborasi, serta relasi-relasi diagram. Diagram

ini umumnya dijumpai pada permodelan sistem berorientasi objek. Meskipun bersifat statis, sering pula diagram kelas membuat 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 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 penting terutama dalam memperlihatkan sifat dinamis dari antarmuka (*interface*), kelas, kolaborasi dan terutama penting pada permodelan 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 penting terutama dalam permodelan fungsi-fungsi suatu sistem dan memberi tekanan pada aliran kendali antar objek.

8. Diagram komponen (*Component Diagram*) bersifat statis. Diagram komponen ini memperlihatkan organisasi serta ketergantungan sistem / perangkat lunak pada komponen-komponen yang telah ada sebelumnya. Diagram ini berhubungan dengan diagram kelas dimana komponen dipetakan kedalam satu atau lebih kelas-kelas, antarmuka-antarmuka serta kolaborasi-kolaborasi.
9. Diagram *Deployment* (*Deployment Diagram*) bersifat statis. Diagram ini memperlihatkan konfigurasi saat aplikasi dijalankan (*run time*). Memuat simpul-simpul beserta komponen-komponen yang ada didalamnya. *Diagram Deployment* berhubungan erat dengan diagram komponen didalam 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*)

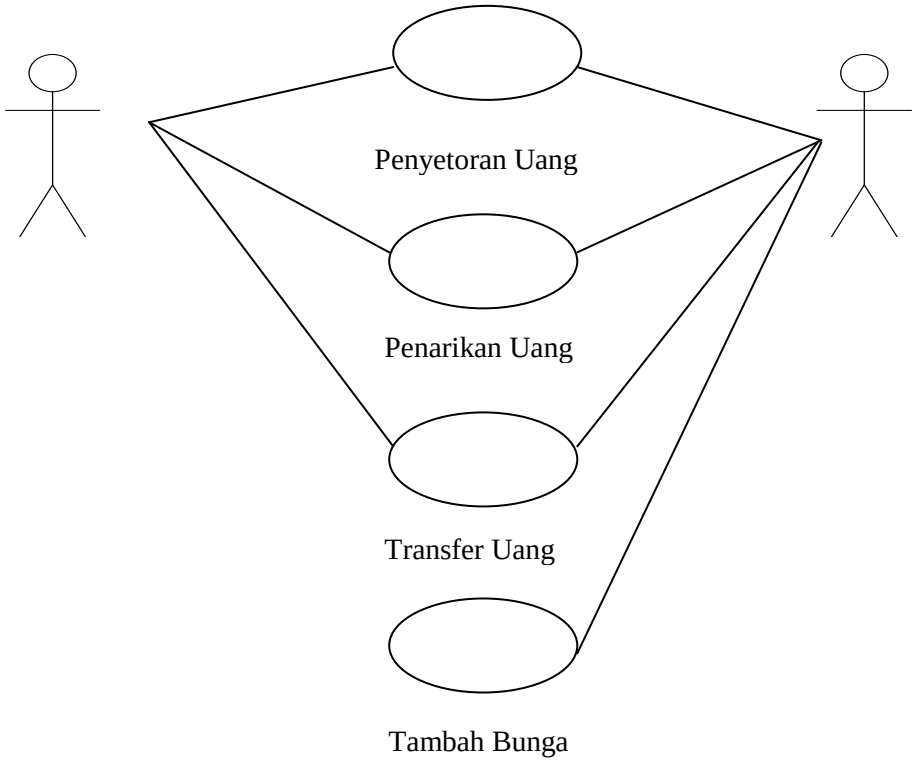
Use Case menggambarkan *external view* dari sistem yang akan kita buat modelnya. Menurut pooley (2005 : 15) mengatakan bahwa model *use case* dapat dijabarkan dalam diagram, tetapi yang perlu diingat, diagram tidak identik dengan model karena model lebih luas dari diagram.

Komponen pembentuk diagram *use case* adalah :

1. Actor (*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 dibawah ini merupakan salah satu contoh bentuk diagram *use case*.

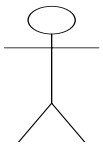


Gambar II.2. : Diagram Use Case

Sumber : Prabowo Pudjo Widodo ; 2011 : 17

a. Aktor

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

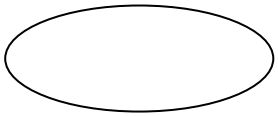


Gambar II.3. : Aktor

Sumber : Prabowo 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 : Prabowo 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 *behavior* (perilaku), jadi seharusnya dalam frase kata kerja. Untuk membuat namanya lebuh detail 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 membutuhkan *use case* yang membatalkan tujuan, misalnya pada *use case* pemesanan kamar, dibutuhkan pula *use case* pembatalan pemesanan 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)

Biodata
+ Nim : String + Alamat : String + AlamatOrangTua : String + JumlahKakak : Integer + NoTelepon : String + Jurusan : String + Agama : String + NamaAyah : String + Status : String + NoHandphone : String + JumlahAdik : Integer + NamaIbu : String +Tempat/Tgllahir : String

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 sistem dari pada bagaimana sistem dirakit. Diagram ini tidak hanya memodelkan *software* melainkan memodelkan bisnis juga. Diagram aktivitas menunjukkan aktivitas sistem dalam kumpulan aksi-aksi. Ketika digunakan dalam permodelan *software*, diagram aktivitas mempresentasikan pemanggilan suatu fungsi tertentu misalnya *call*. Sedangkan bila digunakan dalam permodelan bisnis, diagram ini menggambarkan aktivitas yang dipicu oleh kejadian-kejadian internal misalnya penggajian setiap jumat sore (Prabowo Pudji Widodo ; 2011 : 143 – 145).

Aktivitas merupakan kumpulan aksi-aksi. Aksi-aksi melakukan langkah sekali saja tidak boleh dipecah menjadi beberapa langkah-langkah lagi.

Contoh aksinya yaitu :

a. Fungsi Matematika

b. Pemanggilan Perilaku

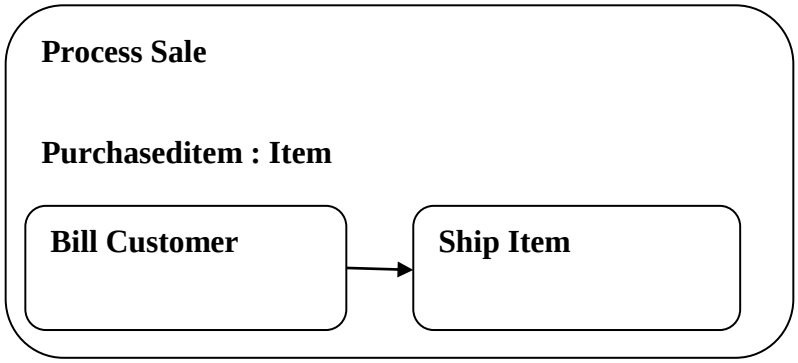
c. Pemrosesan Data

Ketika kita menggunakan diagram aktivitas untuk memodelkan perilaku suatu *classfier* dikatakan konteks 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 sederhana tanpa rincian

Sumber : Prabowo Pudjo Widodo ; 2011 : 145

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



Gambar II.8. Aktivitas dengan detail rincian

Sumber : Prabowo Pudjo Widodo ; 2011 : 145

5. *Sequence Diagram*

Menurut Jhon satzinger, 2010, dalam buku *System Analisis and Design in a Changing World*, "System Sequence Diagram (SSD) adalah diagram yang digunakan untuk mendefenisikan input dan output serta urutan interaksi antara pengguna dan system untuk sebuah use case (E.Triandini dan G. Suardika ; 2012 : 71).