BAB II

TINJAUAN PUSTAKA

II.1. Pengertian Sistem

Secara leksikal, sistem berarti susunan yang teratur dari pandangan, teori, asas dan sebagainya. Dengan kata lain, sistem adalah suatu kesatuan usaha yang terdiri dari bagian-bagian yang berkaitan satu sama lain yang berusaha mencapai suatu tujuan dalam suatu lingkungan kompleks. Pengertian tersebut mencerminkan adanya beberapa bagian dan hubungan antara bagian, ini menunjukkan kompleksitas dari sistem yang meliputi kerja sama antara bagian yang interpenden satu sama lain. Selain itu dapat dilihat bahwa sistem berusaha mencapai tujuan. Pencapaian tujuan ini menyebabkan timbulnya dinamika, perubahan-perubahan yang terus menerus perlu dikembangkan dan dikendalikan.

Definisi tersebut menunjukkan bahwa sistem sebagai elemen-elemen yang saling berinteraksi secara teratur dalam rangka mencapai tujuan atau sub tujuan (Marimin ; 2008 : 1).

II.2. Pengertian Informasi

Informasi merupakan hasil pengolahan data atau fakta yang dikumpulkan dengan cara tertentu. Informasi disajikan dalam bentuk yang mudah dipahami dan merupakan pengetahuan yang relevan yang dibutuhkan untuk menambah wawasan bagi pemakainya guna mencapai suatu tujuan.

Sebagai contoh, data dapat berupa nama karyawan, jumlah jam kerja, dan lain sebagainya. Jika banyaknya jam kerja dikalikan dengan besarnya upah per

jam, maka akan diperoleh gaji kotor. Jika seluruh gaji kotor tersebut dijumlahkan, maka akan diperoleh total gaji kotor. Setelah pemrosesan dilakukan terhadap data, maka akan diperoleh informasi yang dapat mengungkapkan tentang gaji kotor per karyawan dan total biaya gaji yang harus disediakan oleh perusahaan. (Budi Sutedjo Dharma Oetomo ; 2006 : 12).

II.3. Pengertian Sistem Informasi

Sistem informasi adalah sebagai kumpulan elemen yang saling berhubungan satu sama lain yang membentuk satu kesatuan untuk mengintegrasikan data, memproses dan menyimpan serta mendistribusikan informasi. Dengan kata lain, sistem informasi merupakan kesatuan elemen-elemen yang saling berinteraksi secara sistematis dan teratur untuk menciptakan dan membentuk aliran informasi yang akan mendukung pembuatan keputusan dan melakukan kontrol terhadap jalannya perusahaan. (Budi Sutedjo Dharma Oetomno ; 2006 : 11)

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

Kamus data (*data dictionary*) mencakup definisi-definisi dari data yang disimpan dalam basis data dan dikendalikan oleh sistem manajemen basis data. Struktur basis data yang dimuat dalam kamus data adalah kumpulan dari seluruh definisi field, definisi tabel, relasi tabel dan hal-hal lainnya. Nama field data, jenis data (seperti teks dan 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 mempergunakan data tidak akan ikut terpengaruh (Raymond McLeod Jr dan George P Schell ; 2007 ; 171).

II.6. Normalisasi

Menurut Kusrini (2007 : 39 – 43), 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 (3rd normal form)

Sebuah tabel dikatakan memenuhi bentuk normal ketiga (3NF), jika untuk setiap ketergantungan fungsional dengan notasi X -> A, dimana A mewakili semua atribut tunggal di dalam tabel yang tidak ada di dalam X, maka :

- X haruslah *superkey* pada tabel tersebut.
- Atau A merupakan bagian dari *primary key* pada tabel tersebut.

e. Bentuk Normal Tahap Keempat dan Kelima

Penerapan aturan normalisasi sampai bentuk normal ketiga sudah memadai untuk menghasilkan tabel berkualitas baik. Namun demikian, terdapat pula bentuk normal keempat (4NF) dan kelima (5NF). Bentuk Normal keempat berkaitan dengan sifat ketergantungan banyak nilai (*multivalued dependency*) pada suatu tabel yang merupakan pengembangan dari ketergantungan fungsional. Adapun bentuk normal tahap kelima merupakan nama lain dari *Project Join Normal Form* (PJNF).

f. Boyce Code Normal Form (BCNF)

- Memenuhi 1st NF
- Relasi harus bergantung fungsi pada atribut superkey.

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. Sekilas Tentang Visual Basic

Visual basic 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 :

a. *Data Compression*
b. *Change Data Capture*
c. *Filtered Indexes*
d. *Table-Valued Parameter*
e. *Sparse Column*
f. *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 mensupport 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 umum 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 ke dalam satu atau lebih kelas-kelas. Antarmuka-antarmuka serta kolaborasi-kolaborasi.
9. Diagram *Deployment* (*Deployment Diagram)* bersifat statis. Diagram ini memperlihatkan konfigurasi saat aplikasi dijalankan (*run time)*. Memuat simpul-simpul berserta komponen-komponen yang ada di dalamnya. Diagram *Deployment* berhubungan erat dengan diagram komponen 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.

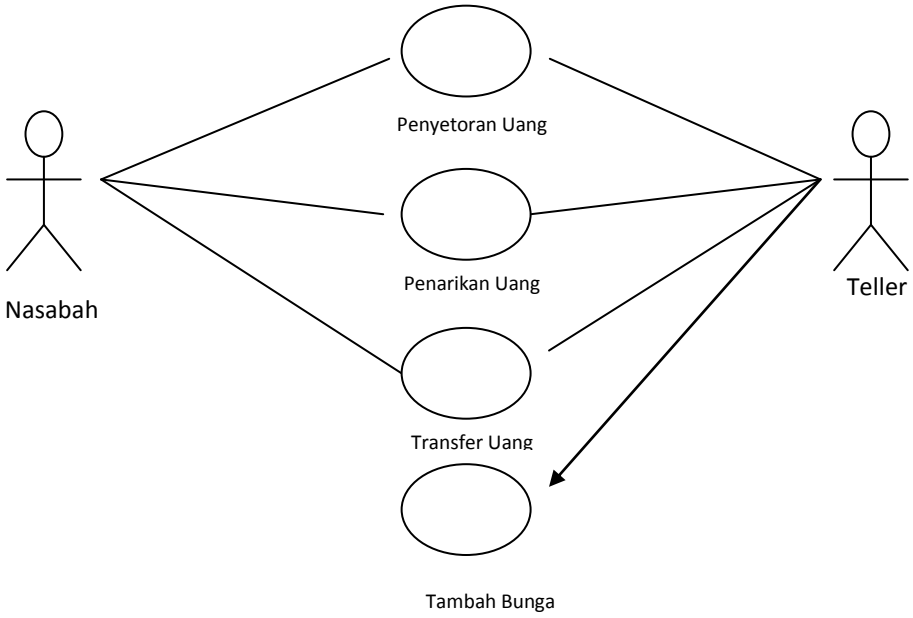
Diagram Use Case (use case diagram)

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*.

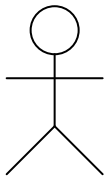


Gambar II.1. Diagram *Use Case*

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*.

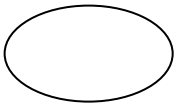


Gambar II.2. Aktor

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

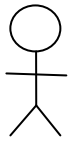
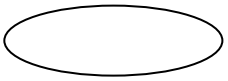
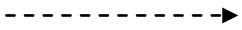
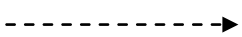


Gambar II.3. Simbol *Use Case*

Sumber : Probowo Pudjo Widodo (2011:22)

Berikut Simbol-simbol dalam *Use Case*, yaitu :

Tabel II.1 Tabel *Simbol-simbol Use case*

NAMA	KETERANGAN	SIMBOL
Actor	Seseorang atau sesuatu yang berinteraksi dengan sistem yang sedang kita kembangkan .	
Use Case	Peringkat Tertinggi dari fungsional yang dimiliki sistem. .	
Relasi Asosiasi	Relasi yang terjadi antara aktor dengan use case biasanya berupa asosiasi. .	
Include Relationship	Relasi cakupan memungkinkansuatu use case untuk menggunakan fungsionalitas yang disediakan oleh use case lainnya.	<<Include> 
Extends Relationship	Memungkinkan suatu use case memiliki kemungkinan untuk memperluas fungsional yang disediakan use case yang lainnya.	<<Extends> 

Sumber : Adi Nugroho(2005:49)

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 us*e case* pemesanan kamar, dibutuhkan pula *use case* pembatalan pesanan kamar.

e. Batasi use case hingga satu perilaku saja.

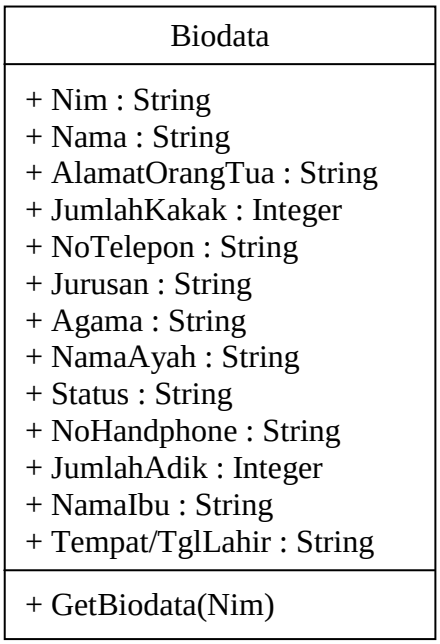
Kadang kita cenderung membuat *use case* yang lebih dari satu tujuan aktivitas. Guna menghindari kerancuan, jagalah use case kita hanya fokus pada satu hal. Misalnya, penggunaan *use case check in* dan *check out* dalam satu *use case* menghasilkan ketidakfokusan, karena memiliki dua perilaku yang berbeda.

3. Diagram Kelas (*Class Diagram*)

Diagram kelas 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.4. Notasi *Domain Diagram Class*

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



Gambar II.5. 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).

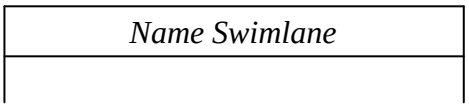
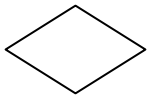
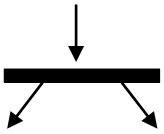
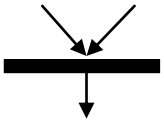
Aktivitas merupakan kumpulan aksi-aksi. Aksi-aksi nelakukan langka sekali saja tidak boleh dipecah menjadi beberapa langkah-langkah lagi. Contoh aksinya yaitu :

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

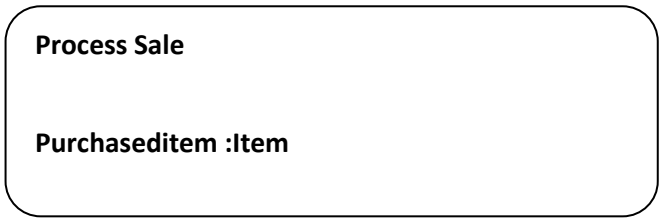
Ketika kita menggunakan diagram aktivitas untuk memodelkan perilaku suatu *classfier* dikatakan kontek dari aktivitas. Aktivitas dapat mengakses atribut dan operasi *classfier*, tiap objek yang terhubung dan parameter-parameter jika aktivitas memiliki hubungan dengan perilaku. Ketika digunakan dengan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

Berikut simbol-simbol dalam *Activity Diagram* yaitu:

Tabel II.2. Tabel Simbol-simbol *Activity Diagram*

Simbol	Keterangan
Status Awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
Swimlane 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.
Aktivitas 	Aktivitas yang dilakukan sistem , aktivitas biasanya diawali dengan kata kerja.
Percabangan/ *decision*	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
Penggabungan / *join* 	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
Fork 	Digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel.
Join 	Digunakan untuk menunjukkan kegiatan yang digabungkan.
Status Akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.

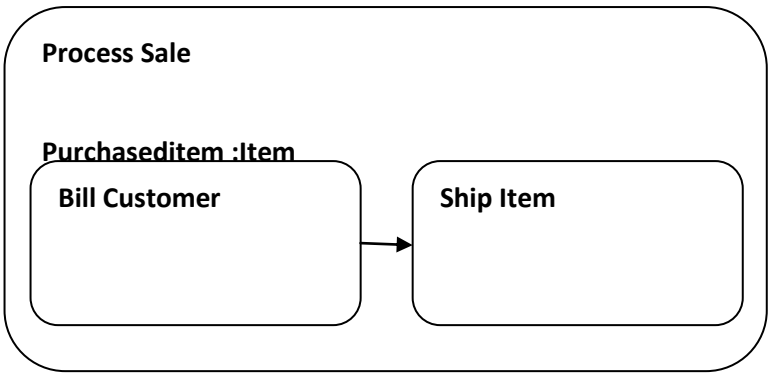
Sumber : Adi Nugroho (2005 : 89)



Gambar II.6. Aktivitas serderhana tanpa rincian

Sumber : Probowo Pudjo Widodo (2011:145)

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



Gambar II.7. Aktivitas dengan detail rincian

Sumber : Probowo Pudjo Widodo (2011:145)

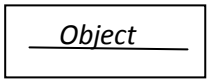
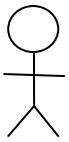
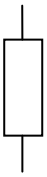
5. *Sequence Diagram*

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

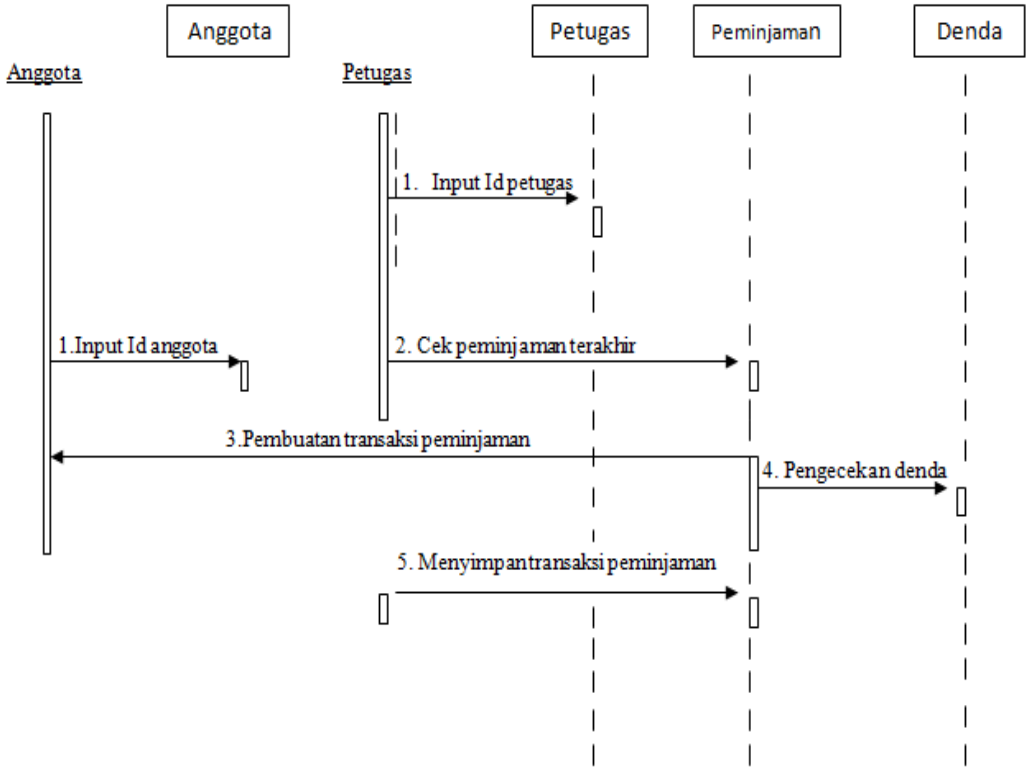
interaksi antara pengguna dan sistem untuk sebuah use case (E. Triandini dan G. Suardika ; 2012 : 71).

Berikut adalah notasi-notasinya :

Tabel II.3. Notasi Sequence Diagram

Object	*Object* merupakan instance dari sebuah *class* dan dituliskan tersusun secara horizontal. Digambarkan sebagia sebuah *class* (kotak) dengan nama obyek di dalamnya yan g diawali dengan sebuah titik koma.	
Actor	*Actor* juga dapat berkomunikasi dengan objek, maka *actor* juga dapat diurutkan sebagai kolom. Simbol *Actor* sama dengan simbol pada *Actor Use Case Diagram.*	
Lifeline	*Lifeline* mengindikasikan keberadaan sebuah object dalam basis waktu. Notasi untuk *Lifeline* adalah garis putus-putus vertikal yang ditarik dari sebuah obyek.	
Activation	*Activation* dinotasikan sebagai kotak segi empat yang digambar pada sebuah *Lifeline. Activation* mengindikasikan sebuah obyek yang akan melakukan sebuah aksi.	
Message	*Message* digambarkan dengan anak panah horizontal antara *activation. Mesaage* mengindikasikan komunikasi antara *object-object.*	

Sumber : Adi Nugroho (2005 : 92)



Gambar II.8. Notasi Sequence Diagram

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

Keterangan :

1. Anggota memasukkan id pada sistem
2. Petugas memasukkan id petugas
3. Petugas melakukan pengecekan pada peminjaman buku terakhir
4. Petugas membuat transaksi peminjaman buku
5. Petugas melakukan pengecekan denda
6. Petugas melakukan penyimpanan ke transaksi peminjaman

II.11. Surat Ketetapan Pajak Daerah (SKPD)

Surat Ketetapan Pajak Daerah yang disingkat SKPD adalah surat ketetapan pajak yang menentukan besarnya jumlah pokok pajak yang terutang terutama denda administrasi kepada Wajib Pajak (WP). Biasanya di dalam selembar Surat Ketetapan Pajak Daerah berisi tentang PKB/BBN-KB dan SWDKLLJ. Dan Draft SKPD adalah Surat Ketetapan Pajak Daerah sementara, sebagai dasar Wajib Pajak untuk melakukan pembayaran atas pajak yang terhutang. Sedangkan SPPKB (Surat Pemberitahuan Pajak Kendaraan Bermotor) yang digunakan oleh wajib pajak adalah SPPKB yang disediakan oleh Dinas Pendapatan berlogokan Pemprovsu sebelah kiri dan Nomor Seri sebelah kanan melalui *security printing*.

Berikut ini ketetapan pajak dan pembayaran pajak dapat di lihat sebagai berikut :

1. Berdasarkan SPPKB, pajak ditetapkan dengan menerbitkan SKPD atau dokumen lain.
2. Pembayaran pajak dapat dilakukan pada saat pendaftaran.
3. Pemilik Kendaraan Bermotor yang telah membayar lunas pajaknya diberikan tanda pelunasan pajak (SKPD) yang telah di validasi.
4. Pembayaran pajak diterima oleh petugas kasir atau petugas yang ditunjuk sesuai dengan ketentuan yang berlaku sebagai pembantu bendahara khusus penerima UPT Dinas Pendapatan Daerah.
5. Masa pajak adalah 12 tahun berturut-turut yang merupakan tahun pajak dimulai pada saat pendaftaran kendaraan bermotor.

Sumber : Dokumen SAMSAT Medan Utara (2012 : 6)