BAB II

TINJAUAN PUSTAKA

II.1. Pengertian Perancangan

Perancangan sistem adalah penentuan proses data yang diperlukan oleh sistem baru. Jika sistem itu berbasis komputer, rancangan dapat menyertakan spesifikasi jenis peralatan yang digunakan. tahap-tahap perancangan sistem informasi adalah sebagai berikut :

a. Menyiapkan rancangan sistem yang terinci
b. Mengidentifikasi berbagai alternatif konfigurasi sistem
c. Mengevaluasi berbagai alternatif konfigurasi sistem
d. Memilih konfigurasi terbaik e. Menyiapkan usulan penerapan
e. Menyetujui atau menolak penerapan sistem

System design is the specification or construction of a technical , computer based solution for the business requirements identified in a system analysis". Yang diterjemahkan sebagai berikut: "Perancangan sistem adalah spesifikasi atau perwujudan dari solusi teknis berbasiskan komputer untuk kebutuhan bisnis yang diidentifikasikan di sistem analisis". Menurut Romney dan Steinbart (2006, p792): "*System design is the process of preparing detail specifications for development of a new information system*". Yang diterjemahkan sebagai berikut: "Perancangan sistem adalah suatu proses detail spesifikasi untuk mengembangkan sebuah sistem informasi yang baru". Berdasarkan pendapat-pendapat diatas, dapat disimpulkan bahwa perancangan sistem adalah proses

mengimplementasikan hasil-hasil dari analisis sistem ke dalam suatu rancangan sistem yang baru (Henny Hendarti ; 2009 : 158).

Model perancangan sesungguhnya adalah modal objek yang mendeskripsikan realisasi fisik *use case* dengan cara berfokus pada bagaimana spesifikasi-spesifikasi kebutuhan fungsional dan *non*-fungsional, bersama dengan batasan-batasan lain yang berhubungan dengan lingkungan implemenatasi, memiliki imbas langsung pada pertimbangan-pertimbangan pada aktivitas-aktivitas yang dilakukan pada tahap implementasi. Tambahannya, model perancangan sesungguhnya secara langsung bertindak sebagai abstraksi implementasi sistem/perangkat lunak dan dengan sendirinya model perancangan suatu saat nanti akan menjadi asupan bagi aktivitas-aktivitas selanjutnya yang kelak akan terdefinisi pada tahap implementasi (Adi Nugroho ; 2010 : 212).

II.2. Pengertian Aplikasi

Program aplikasi adalah program siap pakai atau program yang direka untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain. Aplikasi juga diartikan sebagai penggunaan atau penerapan suatu konsep yang menjadi pokok pembahasan atau sebagai program komputer yang dibuat untuk menolong manusia dalam melaksanakan tugas tertentu. Aplikasi software yang dirancang untuk penggunaan praktisi khusus, klasifikasi luas ini dapat dibagi menjadi 2 (dua) yaitu:

a. Aplikasi *software* spesialis, program dengan dokumentasi tergabung yang dirancang untuk menjalankan tugas tertentu.
b. Aplikasi paket, suatu program dengan dokumentasi tergabung yang dirancang untuk jenis masalah tertentu (Rahmatillah ; 2011: 3).

II.3. Pengertian Surat

Sedangkan pengertian surat adalah sehelai kertas atau lebih yang digunakan untuk mengadakan komunikasi secara tertulis. Surat adalah pernyataan tertulis dari pihak satu kepihak lain, atas nama perseorangan ataupun atas nama jabatan. Dari beberapa pendapat diatas maka dapat disimpulkan mengenai pengertian surat yaitu sarana atau wahana komunikasi tertulis yang ditujukan kepada orang lain atau suatu instansi dengan tujuan untuk menyampaikan suatu hal baik itu berupa informasi, perintah atau sebuah pemberitahuan.

Surat masuk adalah semua jenis surat yang diterima dari instansi lain maupun perorangan, baik yang diterima melalui pos, maupun yang diterima dari kurir dengan mempergunakan buku pengiriman.

Surat keluar adalah surat yang lengkap (bertanggal, bernomor, berstempel dan ditandatangani oleh pejabat yang berwenang) yang dibuat oleh suatu instansi atau lembaga lain. Surat keluar biasanya dikirim melalui pos atau kurir.

Dari pengertian-pengertian di atas dapat disimpulkan bahwa prosedur pengelolaan surat masuk dan surat keluar adalah pekerjaan surat-menyurat yang harus dilakukan secara tertata dan berurutan dengan kegiatan utama yaitu mengelola, mengatur dan mengurus surat-menyurat agar dapat memperlancar administrasi instansi tersebut (Indra Widyantoko ; 2013 : 19-20).

II.4. *Client Server*

Aplikasi *database client/server* terdiri dari dua bagian, yaitu *server* dan *client. Server,* yang sering disebut sebagai *back-end,* akan berhubungan dengan media penyimpanan data/*storage device*. Tugasnya adalah mengatur agar semua data dapat tersimpan dengan benar pada media

penyimpanannya. *Server* menyediakan fungsi-fungsi untuk melakukan proses *inquiry* maupun manipulasi data. Berhubung *server* ini berhubungan sangat erat dengan data, maka banyak orang menyebutnya sebagai *database server*. Oleh karena fungsi-fungsi pada *server* sangat rumit, sangat disarankan agar kita membeli dari *vendor-vendor* berpengalaman; jadi kita tidak perlu membuat sendiri. Bagian lain dari aplikasi *database client/server* adalah *client. Client*, yang sering disebut sebagai *front-end,* berinteraksi dengan para pemakai aplikasi. Fungsi dari *client* ini sangatlah bervariasi, perlu disesuaikan dengan keperluan para pemakainya. Dalam melaksanakan fungsinya, *client* akan membuat hubungan ke *database server* dan memanggil fungsi-fungsi yang tersedia pada *server*.

Sebuah *database server* dapat dihubungkan dengan lebih dari satu *client*. Bisa jadi, *client-client* tersebut memanggil fungsi-fungsi yang sama dalam satu waktu. *Server* mengatur bagaimana cara melaksanakan fungsi-fungsi itu sehingga data yang di proses dapat selalu dalam keadaan benar.

Ada banyak database server yang beredar di pasaran dengan harga dan fitur yang berbeda-beda. Kalau anda menginkan database server yang murah meriah, anda boleh mencoba *MySQL*. Kalau anda menginkan database server yang memiliki banyak fasilitas dan kat menampung data dalam jumlah besar, anda boleh mencoba *Microsoft SQL* atau *Oracle*. Database server yang akan dipakai dalam bab-bab selanjutnya adalah Interbase. Sebelum melanjutkan, pastikan anda telah menginstal interbase Server pada komputer. Apabila belum, anda dapat menemukan petunjuk insalasinya pada bab pertama buku ini. Pastikan juga bahwa anda sudah menghasilkan code tersebut dalam pemrograman kita. (David Ciang ; 2007 : 24).

II.4.1.Konsep *Client Server*

Konsep *client server* yang berbasis pada aturan bahwa komputer *server* hanya akan mengirim data yang dibutuhkan oleh *workstation/client,* di mana proses penyiapan data dilakukan pada komputer *server*. Proses tersebut sedikit banyak dapat mengurangi beberapa permasalahan, baik dari segi lalu lintas data maupun sumber daya dan biaya komputerisasi, karena kini sebuah perusahaan dapat menggunakan komputer berkemampuan rendah sebagai *workstation* dan memberi alokasi dana lebih besar untuk memperoleh komputer *server* dengan kemampuan lebih baik. Disamping itu kemampuan data lebih terjamin, salah satu contoh paling populer dari konsep *client server* adalah sistem jaringan internet, di mana dengan menggunakan komputer sederhana atau lewat ponsel, kita dapat mengakses data pada jaringan komputer lainnya (Harip Santoso ; 2007 : 2).

II.4.2.Arsitektur *Client Server*

Berdasarkan pada cara PC *Client* dihubungkan ke komputer *server*, dikenal dua macam tingkatan arsitektur yaitu model Dua Tiered dan Model Tiga Tiered. Dua Tier/Dua Tingkatan adalah proses di mana komputer *workstation* membuat hubungan ke komputer *server* dan mempertahankan hubungan tersebut sampai proses selesai. Misalnya pada proses pemasukan data atau membuat sebuah laporan. Hubungan yang terbentuk sangat konsumtif dalam sumber daya (alokasi memori, ruang *harddisk*, kontrol, dan lainnya), karenanya tidak efektif jika dilakukan untuk hubungan yang melibatkan banyak pemakai (contohnya pada sistem jaringan internet). Oleh karena itu tidak heran jika konsep Dua Tingkatan hanya pengguna digunakan model Tiga Tiered.

Tiga Tier/Tiga Tingkatan adalah model proses di mana kita menambahkan sebuah komputer yang bertugas untuk berhubungan dengan komputer *server* (SQL *Server*) sehingga *workstation/client* tidak dapat berhubungan langsung dengan komputer *server (*SQL *Server*).

Contoh paling populer dari model Tiga Tier adalah sistem jaringan internet, di mana semua permintaan *client* akan ditangani oleh komputer web *server* dan dari web *server* hubungan ke SQL *server* dilakukan. Model ini cukup efektif, ia dapat membantu meningkatkan keamanan data karena begitu permintaan data ke SQL *Server* selesai dilakukan (informasi/tabel data di copy-kan ke komputer web *server*), hubungan ke SQL *server* diputus (Harip Santoso ; 2007 : 3).

II.5. *Visual Basic*

Visual Basic dibuat Microsoft, merupakan salah satu bahasa pemrograman berorientasi objek yang mudah dipelajari. Selain menawarkan kemudahan, Visual Basic juga cukup andal untuk digunkan dalam pembuatan berbagai aplikasi, terutama aplikasi database. Visual Basic Merupakan bahasa perograman *event drive*, di mana program akan menunggu samapai aa respons dari user/pemakai program aplikasi yang dapat berupa kejadian atau event, misalnya ketika user mengklik tombola tau menekan enter. Jika kita membuat aplikasi dengan Visual Basic maka kita akan mendapatkan file yang menyusun aplikasi tersebut, yaitu :

1. File Project (*.vbp)

   File ini merupakan kumpulan dari aplikasi yang kita buat, file project bisa berupa file*.frm,*.dsr atau file lainnya.

2. File From (*.frm)

   File ini merupakan file yang berfungsi untuk menyimpan informasi tentang bentuk form maupun interface yang kita buat.

Untuk menjalankan Visual Basic sanagt mudah :

1. Dari menu Start, pilih program selanjutnya pilih Microsoft Visual Basic 6.0 jika anda menggunakan Visual Basic Versi 6.0.

2. Ketika pertama kali dijalankan maka akan muncul kotak dialog yang memliki 3 tabulasi yaitu :
   a. New, digunakan untuk membuat project baru
   b. Exiting, untuk membuka project yang pernah kita buat.
   c. Recent, untuk membuka project yang pernah kita buka.(Edy Winarno, 2010 : 83).

II.6. Pengertian Database

Database adalah sekompulan data mentah yang disusun menurut logika tertentu dan terorganisasi dalam bentuk yang dapat disimpan dan diproses oleh komputer. contoh database dapat berisi data pegawai, data penjualan, pembayaran, dan lain-lain. data internal dari akunting, keuangan, penjualan dan bidang-bidang bisnis lainnya yang disimpan dalam suatu sistem komputer dan disusun menurut logika tertentu disebut sebagai *internal database*.

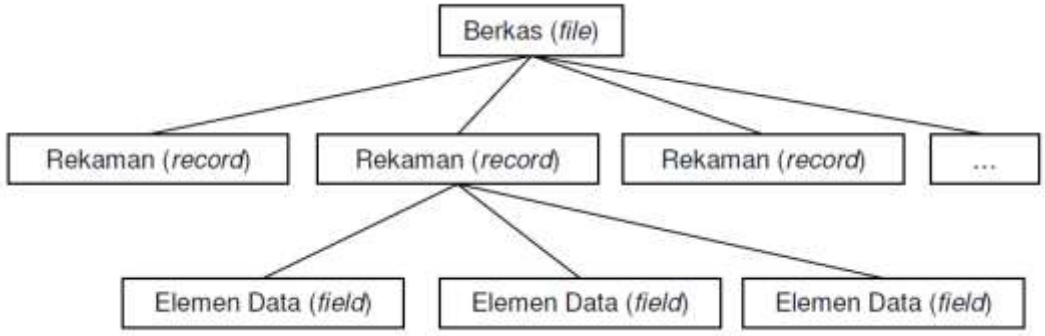
Database seringkali disimpan dalam suatu perangkat tertentu pada komputer, seperti *hard disk*, *compact disk*, dan sebagainya. hubungan antar sistem *database* dan sistem *software* sangat kuat karena sistem database yang dipakaiu sangat menentukan kemudahan aksesnya data sementara *software* sendiri memungkinkan peneliti memanipulasi data untuk dianalisis (Dermawan Wibisono ; 2008 : 129).

II.6.1. Hierarki Database

Data diorganisasikan ke dalam bentuk elemen data (*field*), rekaman (*record*), dan berkas (*file*). Definisi dari ketiganya adalah sebagai berikut:

Elemen data adalah satuan data terkecil yang tidak dapat dipecah lagi menjadi unit lain yang bermakna. Misalnya data siswa terdiri dari NIS, Nama, Alamat, Telepon atau Jenis

Kelamin. Rekaman merupakan gabungan sejumlah elemen data yang saling terkait. Istilah lain dari rekaman adalah baris atau tupel. Berkas adalah himpunan seluruh rekaman yang bertipe sama (Haidar Dzacko ; 2007 : 1).



Gambar II.1. Hirarki Data

(Sumber : Haidar Dzacko ; 2007 : 1)

II.6.2. Model Database

Model data dapat dikelompokkan berdasarkan konsep pembuatan deskripsi struktur basis data, yaitu:

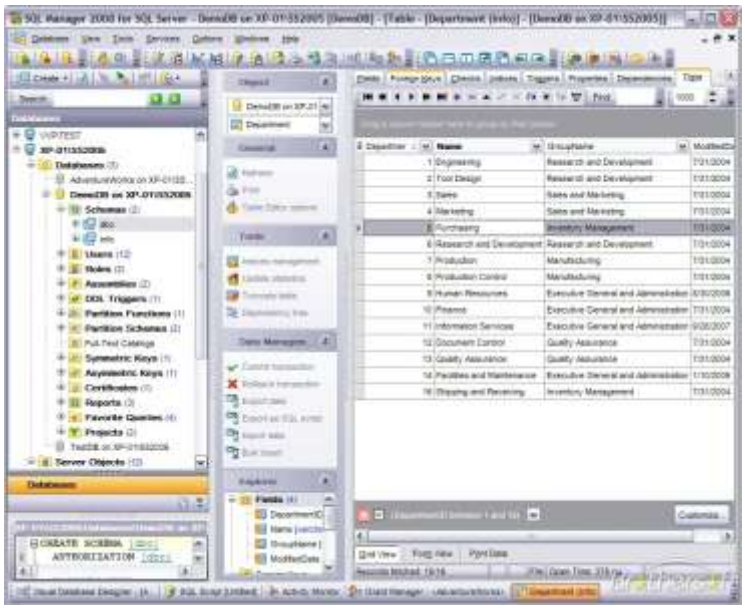
1. Model data konseptual (*high level*) menyajikan konsep tentang bagaimana *user* memandang atau memperlakukan data. Dalam model ini dikenalkan tiga konsep penyajian data yaitu:
   a. *Entity* (entitas) merupakan penyajian obyek, kejadian atau konsep dunia nyata yang keberadaannya secara eksplisit didefinisikan dan disimpan dalam basis data, contohnya Mahasiswa, Matakuliah, Dosen, Nilai dan lain sebagainya.
   b. *Atribute* (atribut) adalah keterangan-keterangan yang menjelaskan karakteristik dari suatu entitas seperti NIM, Nama, Fakultas, Jurusan untuk entitas Mahasiswa.

c. *Relationship* (hubungan) merupakan hubungan atau interaksi antara satu entitas dengan yang lainnya, misalnya entitas pelanggan berhubungan dengan entitas barang yang dibelinya.

2. Model data fiskal (*low level*) merupakan konsep bagaimana deskripsi detail data disimpan ke dalam komputer dengan menyajikan informasi tentang format rekaman, urutan rekaman, dan jalur pengaksesan data yang dapat membuat pencarian rekaman data lebih efisien.
3. Model data implementasi (*representational*) merupakan konsep deskripsi data disimpan dalam komputer dengan menyembunyikan sebagian detail deskripsi data sehingga para *user* mendapat gambaran global bagaimana data disimpan dalam komputer. Model ini merupakan konsep model data yang digunakan oleh model hierarki, jaringan dan relasional (Haidar Dzacko ; 2007 : 3).

II.7. Pengertian SQL Server

SQL Server 2008 adalah sebuah RDBMS (Relational Database Management System) yang di-develop oleh Microsoft, yang digunakan untuk menyimpan dan mengolah data. Pada SQL Server 2008, kita bisa melakukan pengambilan dan modifikasi data yang ada dengan cepat dan efisien. Pada SQL Server 2008, kita bisa membuat object-object yang sering digunakan pada aplikasi bisnis, seperti membuat database, table, funcation, stored procedure, trigger dan view. Selain object, kita juga menjalankan perintah SQL (Structured Query Language) untuk mengambil data. (Cybertron Solution; 2010 : 101-102).



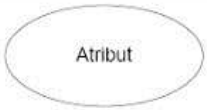
Gambar II.2. Tampilan SQL Server

(*Sumber : Cybertron Solution; 2010 : 101-102*)

II.8. *Entity Relationship Diagram* (ERD)

Entity Relationship Diagram atau ERD adalah alat pemodelan data utama dan akan membantu mengorganisasi data dalam suatu proyek ke dalam entitas-entitas dan menentukan hubungan antarentitas. Proses memungkinkan analis menghasilkan struktur basisdata yang baik sehingga data dapat disimpan dan diambil secara efisien (Janner Simarmata ; 2010 : 67).

Tabel II.1. Simbol ERD

Notasi	Keterangan
	**Entitas**, adalah suatu objek yang dapat diidentifikasi dalam lingkungan pemakai.
	**Relasi**, menunjukkan adanya hubungan di antara sejumlah entitas yang berbeda.
	**Atribut**, berfungsi mendeskripsikan karakter entitas (atribut yg berfungsi sebagai key diberi garis bawah)
	Garis, sebagai penghubung antara relasi dengan entitas, relasi dan entitas dengan atribut.

(Sumber : Janner Simarmata ; 2010 : 67)

II.9. Teknik Normalisasi

Proses normalisasi menyediakan cara sistematis untuk meminimalkan terjadinya kerangkapan data di antara relasi dalam perancangan logikal basis data. Format normalisasi terdiri dari lima bentuk, yaitu:

II.9.1.Bentuk-bentuk Normalisasi

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

Suatu tabel dikatakan sudah 1NF jika telah memenuhi ketentuan sebagai berikut:

- Tidak ada atribut mempunyai nilai berulang atau nilai *array*
- Tidak mempunyai baris yang rangkap

Bentuk unnormal mengijinkan nilai-nilai pada suatu atribut dapat berulang.

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

Relasi dapat dikatakan format normal kedua jika sudah dalam format normal pertama dan diikuti kondisi sebagai berikut:

- *Key* terdiri dari atribut tunggal
- Setiap atribut *non-key* ketergantungan fungsional pada semua *key* atau tidak terjadinya ketergantungan pada *key composite*.

Misalnya tabel UNIV berada dalam normal kedua dengan mengasumsikan DNO sebagai *key*, kecuali CRSE. Jika ditentukan CNO dan SECNO sebagai *key* composite, atribut *nonkey* CNAME tergantung hanya pada CNO, bukan pada SECNO, sehingga CNAME tidak secara ketergantungan fungsional penuh terhadap *key* (CNO, SECNO).

c. Bentuk normal tahap ketiga (3rd normal form)

Relasi dikatakan format normal ketiga jika sudah dalam format normal kedua dan tidak ada ketergantungan transitif di antara atribut. Misalnya tabel STUDNT mempunyai atribut SSNO sebagai *key* (2NF). Ketergantungan transitif terjadi di antara DNO dan COLREG. Saat DNO determinan COLREG tanpa melibatkan *key* SSNO. Contohnya, DNO=’CS’ termasuk COLREG=’Arts/Sc.’ tidak tergantung oleh atribut SSNO, sehingga STUDNT belum termasuk 3NF. Yang menjadi catatan, ketergantungan transitif tidak akan terjadi jika ada ketergantungan fungsional di antara atribut-atribut *non-key* yang melibatkan *key*.

d. Boyce Code Normal Form (BCNF)

BCNF menentukan setiap determinan adalah kunci kandidat (*candidate key*). Misalnya UNIV mempunyai dua determinan yaitu DNO dan DNAME yang merupakan *kunci kandidat* sehingga termasuk ke dalam BCNF. Di lain pihak

CRSLST dalam 3NF tetapi tidak dalam BCNF. Atribut komposisinya (CNO, SECNO, SID, OFRNG) sebagai kunci-kunci kandidat dan tidak ada ketergantungan transitif, sehingga CRSLST termasuk ke dalam 3NF. Namun atribut CNO adalah determinan saat SECNO tergantung penuh secara fungsional terhadap CNO, walaupun CNO bukan kunci kandidat, sehingga CRSLST belum termasuk BCNF.

e. Bentuk Normal Tahap Keempat dan Kelima

Bentuk ini adalah bentuk normal ketiga atau BCNF dengan nilai atribut tidak tergantung pada nilai banyak (*multivalue dependency*). Konsep pada bentuk ini adalah ketergantungan pada gabungan beberapa atribut (*join dependency*) (Haidar Dzacko ; 2007 : 12).

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

Menurut Windu Gata (2013 : 4) Hasil pemodelan pada OOAD terdokumentasikan dalam bentuk *Unified Modeling Language* (UML). UML adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak.

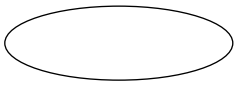
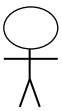
UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem.

Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut :

- *Use case* Diagram

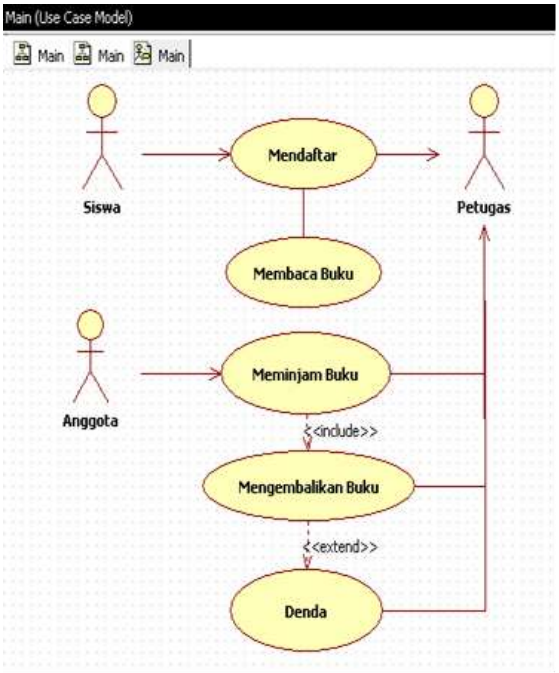
Use case diagram merupakan pemodelan untuk kelakukan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbol-simbol yang digunakan dalam *use case* diagram, yaitu :

Tabel II.2. Simbol *Use Case*

Gambar	Keterangan
	*Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukan pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *use case*.
	Aktor adalah *abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktor, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan *use case*, tetapi tidak memiliki control terhadap *use case*.
	Asosiasi antara aktor dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengidikasikan aliran data.
	Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengidinkasikan bila aktor berinteraksi secara pasif dengan sistem.
	Include, merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program.
	Extend, merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

(Sumber : Windu Gata ; 2013 : 4)

Contoh dari pembuatan *use case diagram* dapat dilihat pada gambar II.2 berikut :



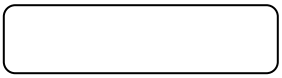
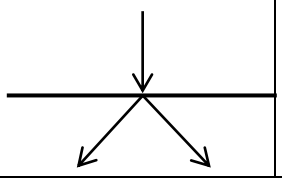
Gambar. II.3. *Use Case Diagram*

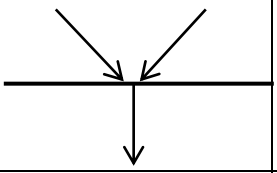
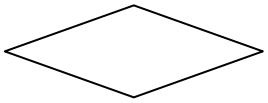
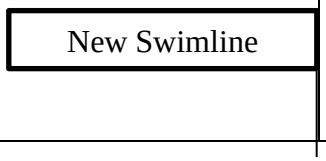
(Sumber : Windu Gata ; 2013 : 4)

- Diagram Aktivitas (*Activity Diagram*)

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram,* yaitu :

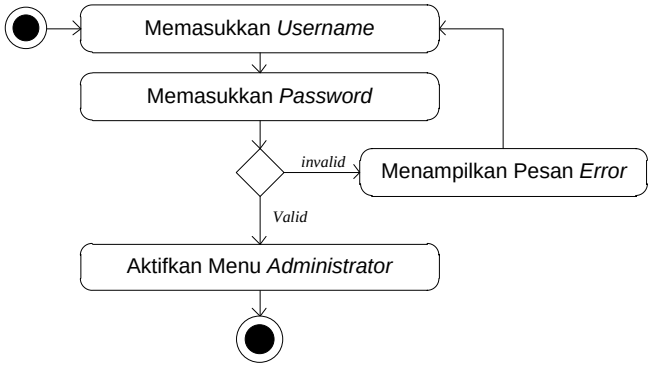
Tabel II.3. Simbol *Activity Diagram*

Gambar	Keterangan
	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*End point*, akhir aktifitas.
	*Activites*, menggambarkan suatu proses/kegiatan bisnis.
	*Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu.

	*Join* (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Points*, menggambarkan pilihan untuk pengambilan keputusan, *true*, *false.*
	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa.

(Sumber : Windu Gata ; 2013 : 6)

Contoh dari pembuatan *activity diagram* dapat dilihat pada gambar II.3 berikut :



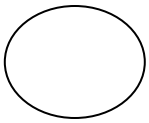
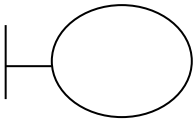
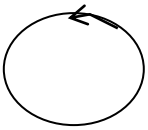
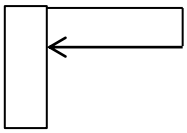
Gambar. II.4. *Activity Diagram*

(Sumber : Windu Gata ; 2013 : 6)

- Diagram Urutan (*Sequence Diagram*)

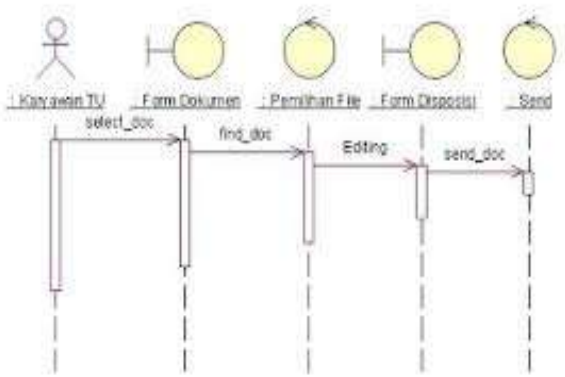
Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Simbol-simbol yang digunakan dalam *sequence diagram*, yaitu :

Tabel II.4. Simbol *Sequence Diagram*

Gambar	Keterangan
	*Entity Class*, merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.
	*Boundary Class*, berisi kumpulan kelas yang menjadi *interface* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan formentry dan *form* cetak.
	*Control class*, suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek.
	*Message*, simbol mengirim pesan antar *class*.
	*Recursive,* menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.
	*Activation*, *activation* mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah operasi.
	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*.

(Sumber : Windu Gata ; 2013 : 7)

Contoh dari pembuatan *sequence diagram* dapat dilihat pada gambar II.4 berikut :



Gambar. II.5. *Sequence Diagram*

(Sumber : Windu Gata ; 2013 : 7)

- *Class Diagram* (Diagram Kelas)

Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggng jawab entitas yang menentukan perilaku sistem.

Class diagram juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan. *Class diagram* secara khas meliputi: Kelas (*Class*), Relasi, *Associations*, *Generalization* dan *Aggregation*, Atribut (*Attributes*), Operasi (*Operations/Method*), *Visibility*, tingkat akses objek eksternal kepada suatu operasi atau atribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau kardinaliti.

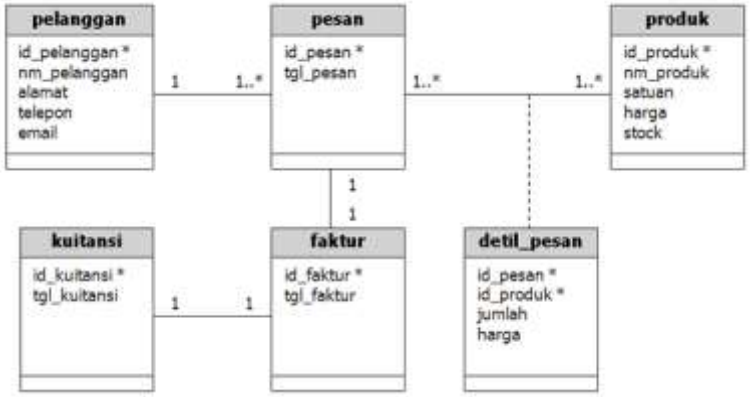
Tabel II.5. *Multiplicity Class Diagram*

Multiplicity	**Penjelasan**

1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimum 4

(Sumber : Windu Gata ; 2013 : 8)

Contoh dari pembuatan *use case diagram* dapat dilihat pada gambar II.5 berikut :



Gambar. II.6. *Class Diagram*

(Sumber : Windu Gata ; 2013 : 8)