BAB II

LANDASAN TEORI

II.1. Pengertian Sistem

Untuk mengawali pembahasan tentang analisis dan perancangan sistem informasi, pemahaman akan sistem terlebih dahulu harus ditekankan. Definisi sistem berkembang sesuai dengan konteks di mana pengertian sistem itu digunakan. Berikut akan diberikan beberapa definisi sistem secara umum : Kumpulan dari bagian-bagian yang bekerja sama untuk mencapai tujuan yang sama. Contoh :

1. Sistem Tatasurya
2. Sistem Pencernaan
3. Sistem Transportasi Umum
4. Sistem Otomatif
5. Sistem Komputer
6. Sistem Informasi

Sekumpulan objek-objek yang saling berelasi dan berinteraksi serta hubungan antar objek bisa dilihat sebagai satu kesatuan yang dirancang untuk mencapai satu tujuan.

Dengan demikian, secara sederhana sistem dapat diartikan sebagai suatu kumpulan atau himpunan dari unsur atau variable-variabel yang saling terogranisasi, saling berinteraksi, dan saling bergantung sama lain. Definisi sistem sebagai seperangkat elemen yang digabungkan satu dengan lainnya untuk satu

tujuan bersama. Sementara definisi sistem dalam kamus *Webster's Unbriged* adalah elemen-elemen yang saling berhubungan dan membentuk satu kesatuan atau organisasi (Hanif Al Fatta ; 2007 : 3).

II.2. Pengertian Informasi

Informasi diartikan sebagai data yang telah diolah dan bermanfaat bagi pemakai dalam rangka pengambilan keputusan. Sebagai ilustrasi, data jumlah jam kerja karyawan, saat data diproses dapat berubah menjadi informasi. Jika jam kerja dikalikan dengan upah per jam, maka didapat hasil pendapatan kotor. Jika pendapatan kotor ini dijumlahkan, maka penjumlahan ini merupakan total biaya gaji karyawan harian. Jumlah biaya gaji ini dapat dijadikan informasi bagi manajemen, misalnya dalam alokasi dana. Jadi informasi merupakan data yang telah diolah dan memiliki arti bagi pemakai (Husein Umar ; 2008 : 190).

II.3. Pengertian Geografis

Pemahaman tentang bumi dimiliki manusia sejak ada di muka bumi ini. Sejak lahir manusia memerlukan berbagai unsur yang ada di bumi. Unsur tersebut seperti udara yang bersih, makanan, pakaian dan pemukiman.

Timbulnya tuntutan pemenuhan berbagai kebutuhan hidup yang tidak diperoleh dari lingkungan tempat tinggalnya dan adanya hasrat keingintahuan tentang benda serta gejala yang ada dipermukaan bumi. Mendorong setiap manusia untuk mengadakan perjalanan ke daerah di luar tempat tinggalnya.

Berkembangnya sistem pengetahuan turut mendorong manusia untuk mengenal alam dan lingkungannya lebih jauh. Misalnya, perdagangan antar

daerah telah mendorong manusia untuk mengenal daerah di luar wilayahnya. Dari hasil kunjungan tersebut, mereka dapat mengenal kondisi alam, penduduk, dan kondisi alam, penduduk dan kondisi lainnya. Berbagai hasil perjalanannya tersebut kemudian disampaikan kepada orang lain sehingga orang lain tertarik untuk mengunjunginya. Berawal dari perjalanan inilah munculnya ilmu geografi (Hartono ; 2008 : 2).

II.4. Sistem Informasi Geografis

Menurut Kenneth C. Laudan di dalam bukunya Sistem Informasi Manajemen (2008 : 167) Sistem Informasi Geografis (*geographic information system-GIS)* adalah kategori khusus dari *DSS* yang menggunakan teknologi visualisasi data untuk menganalisis dan menampilkan data untuk perencanaan dan pengambilan keputusan dalam bentuk peta digital. Peranti lunak tersebut merakit, menyimpan, memanipulasi, dan menampilkan secara geografis informasi referensi, menghubungkan data dengan titik, garis, dan bidang pada sebuah peta. *GIS* mempunyai kemampuan membuat model, memungkinkan manajer untuk mengubah data dan secara otomatis memperbarui *scenario* bisnis untuk mencari solusi yang lebih baik.

GIS membantu pengambilan keputusan yang membutuhkan pengetahuan tentang distribusi penduduk atau sumber daya lain secara geografis. Sebagai contoh, *GIS* mungkin digunakan untuk membantu pemerintah Negara dan pemerintah lokal menghitung waktu respon bahaya untuk bencana alam, untuk membantu perusahaan eceran mengidentifikasi lokasi pertokoan baru, atau

membantu bank mengidentifikasi tempat terbaik untuk membangun cabang atau memasang terminal ATM baru.

II.5. Pengertian *Quantum GIS*

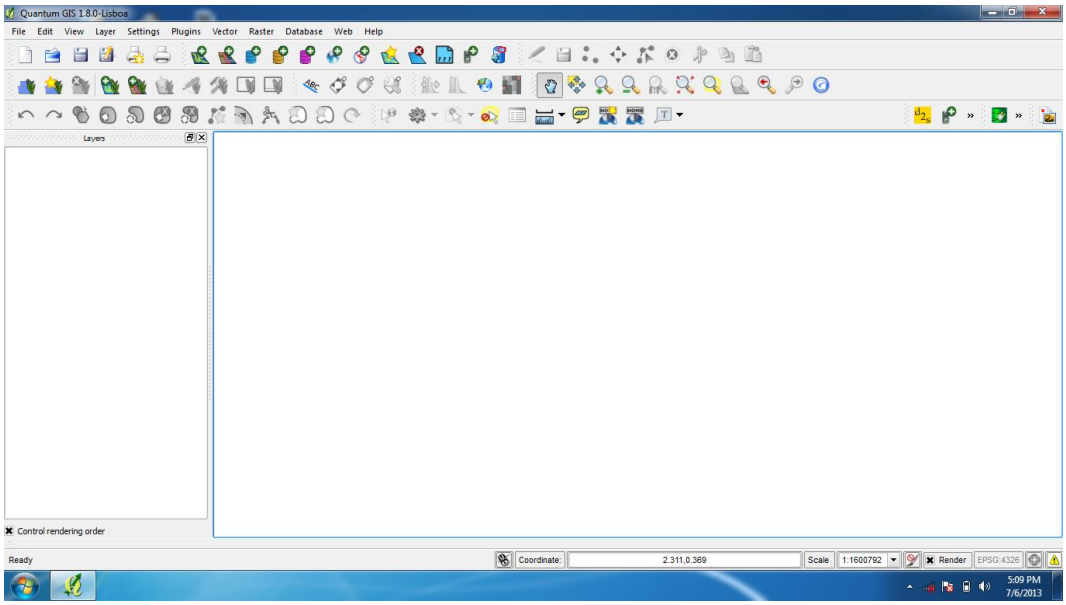
Menurut Prof. Dr. Yupo Chan (2011 : 432) *Quantum GIS* atau yang sering disingkat menjadi *QGIS* adalah sebuah apliasi sistem informasi geografis berbasis desktop yang menyediakan fitur untuk menampilkan data, pengubahan data dan kemampuan dalam menganalisis data spasial. *QGIS* dapat berjalan pada sistem operasi *Linux, UNix, Mac OS,* dan *Windows*.

Quantum GIS dapat dibuat dengan bahasa pemrograman *C++* dan untuk tampilan grafisnya menggunakan pustaka kode *QT-Library. Quantum GIS* memungkinkan untuk membentuk integrasi pada *Plug-In* yang dikembangkan dengan *C++* maupun *Python*.

QT-Library menyediakan tampilan grafis yang dapat berjalan secara *Cross-Platform* dalam Framework pengembangan aplikasi yang didukung oleh perangkat lunak lainnya. *Quantum GIS* memungkinkan untuk dihubungkan atau diintegrasikan dengan berbagai paket perangkat lunak *GIS* yang bersifat *Open-Source* lainnya, seperti *PostGIS, GRASS*, dan *MapServer* untuk memberikan fugsionalitas yang ekstensif kepada penggunanya. *Quantum GIS* secara berkesinambungan terus diperbaiki dan dikembangkan oleh grup pengembang yang aktif dan pengembang sukarela yang secara teratur merilis pembaharuan dan perbaikan pada beberapa kesalahan sistem.

Komponen perangkat lunak *GIS* dibangun berdasarkan blok-blok sehingga dapat ditambahkan perangkat lunak *GIS* dan dibentuk dengan baik serta

lingkungan pengembangan yang dapat disesuaikan untuk pengguna. Fungsi komponen yang spesifik memberikan dedikasi tugas yang ditambahkan pada lingkungan alat pengembangan *GIS*, seperti komponen yang memungkinkan untuk memasukkan format data tertentu agar dapat dikonversi, penganalisis data teratur, dan perangkat pemrosesan citra, perangkat pengembangan pengguna di sisi lainnya sebagai fungsi yang spesifik.

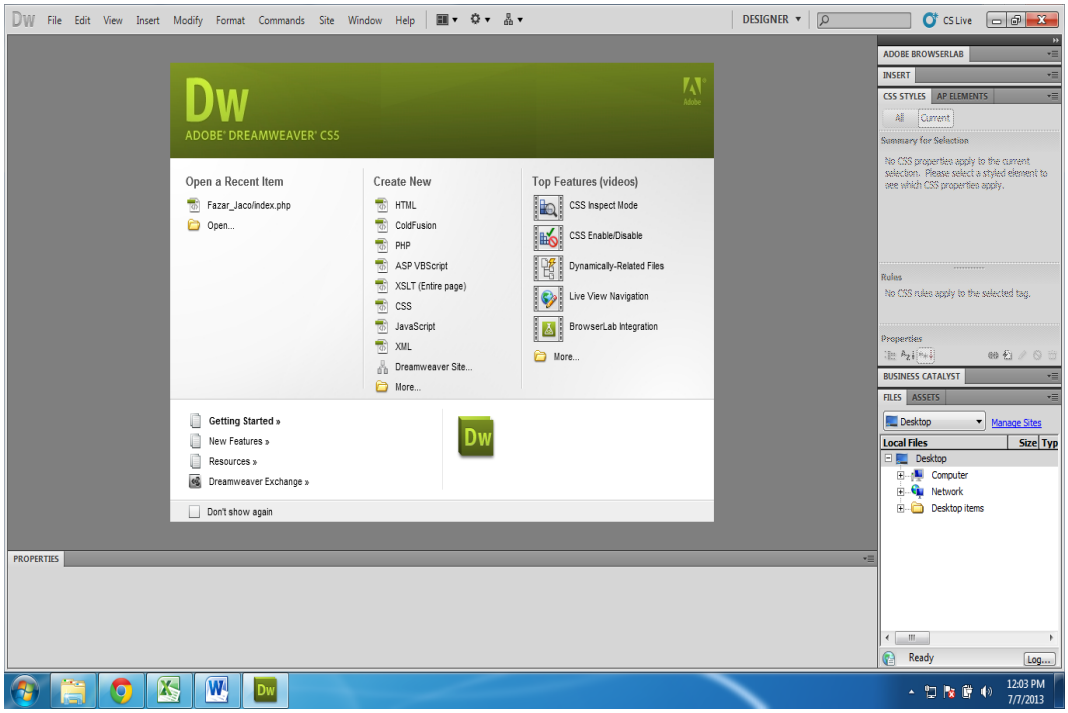


Gambar II.1. Tampilan *Quantum GIS*
(Sumber : Prof. Dr. Yupo Chan 2011 : 432)

II.6. Pengertian *Macromedia Dreamweaver*

Macromedia Dreamweaver adalah sebuah *software web design software web design* yang menawarkan cara mendesain *website* dengan dua langkah sekaligus dalam satu waktu, yaitu mendesain dan memprogram. *Dreamweaver* memiliki satu jendela mini yang disebut *HTML Source*, tempat kode-kode *HTML* tertulis. Setiap kali kita mendesain *web*, seperti menulis kata-kata, meletakkan gambar, membuat tabel dan proses lainnya, *tag-tag HTML* akan tertulis secara

langsung mengiringi proses pengaturan *website*. Artinya kita memiliki kesempatan untuk mendesain. *Website* sekaligus mengenal *tag-tag HTML* yang membangun *website* itu. Di lain kesempatan kita juga dapat mendesain *website* hanya dengan menulis *tag-tag* dan teks lain di jendela *HTML Source* dan hasilnya dapat dilihat langsung di layar (M. Suyanto ; 2009 : 244).



Gambar II.2. Tampilan *Dreamweaver*
(Sumber : M. Suyanto ; 2009 : 244)

II.7. *Haversine Formula*

Haversine Formulla adalah persamaan yang digunakan dalam navigasi, yang memberikan jarak lingkaran besar antara dua titik pada permukaan bola (bumi) berdasarkan bujur dan lintang.

Penggunaan rumus ini mengasumsikan pengabaian efek *ellipsoidal,* cukup akurat untuk sebagian besar perhitungan, juga pengabaian ketinggian bukit dan kedalaman lembah di permukaan bumi (Jurnal, Satria Hidayat ; 2).

Berikut adalah rumus *haversine* Persamaan (1) :

$$d = 2r \sin^1\left(\sqrt{\sin^2\left(\frac{\emptyset 1 - \emptyset 2}{2}\right) + \cos(\emptyset 1)\cos(\emptyset 2)\sin^2\left(\frac{\lambda 2 - \lambda 1}{2}\right)}\right.$$

d = jarak

$\emptyset 2$ = *latitude* awal

$\emptyset 1$ = *latitude* akhir

$\lambda 2$ = *longitude* awal

$\lambda 1$ = *longitude* akhir

Rumus *Haversine* :

d=2r arcsin A

A=

Dimana, d merupakan jarak antara dua titik (sepanjang lingkaran besar dari bola, melihat jarak bola), r merupakan jari - jari bola, phi 2 merupakan nilai *latitude* asal, *phi* 1 merupakan nilai *latitude* tujuan, psi 2 merupakan nilai *longitudinal* asal, dan *psi* 1 adalah nilai *longitudinal* tujuan.

1. Menghitung Jarak Antara Dua Titik Koordinat *Formula Haversine* dipublikasikan oleh R.W. Sinnott pada tahun 1984. *Haversine* merupakan fungsi *trigonometri* yang digunakan untuk melakukan perhitungan jarak

antara dua titik. 6 Berikut ini rumus *Haversine* 7 : R = 6,371km, Δlat = lat2 − lat1, Δlong = long2 − long1, a = sin²(Δlat/2) + cos(lat1) * cos(lat2) * sin²(Δlong/2), c = 2*atan2(√a, √(1−a)), d = R*c.

II.8. Tambal Ban

Menurut Bambang Sumarsono (2014) Perkembangan pengguna kendaraan dari tahun ke tahun semakin meningkat. Pada triwulan-4 tahun 2013, kenaikan produksi kendaraan bermotor, *trailer* dan *semi trailer* naik 11.48%.

1. Kota Medan yang mempunyai 18% dari luas propinsi SUMUT memiliki 3.019.613 unit sepeda motor tahun 2012.
2. Ban bocor bisa terjadi dimana saja dan kapan saja, tidak terkecuali di Kota Medan. Perkembangan pesat dalam pengembangan aplikasi diharapkan dapat membantu pengguna kendaraan bermotor jika terjadi ban bocor. Hal ini didukung dengan penguasaan pasar sebesar 79% pada *kuartal* kedua tahun 2013 dan layanan *Google Maps API* yang disediakan oleh *Google* untuk mengembangkan aplikasi *location-based services*.
3. Ban bocor merupakan kejadian yang tak terduga, Selama ini pengguna kendaraan bermotor yang mengalami ban bocor perlu menanyakan lokasi tambal ban ke warga sekitar untuk mengetahui lokasi tambal ban.

II.9. Pengertian *Database*

Database adalah sekumpulan 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 dipakai sangat menentukan kemudahan aksesnya data sementara *software* sendiri memungkinkan peneliti memanipulasi data untuk dianalisis (Dermawan Wibisono ; 2008 : 129).

II.10. Pengertian *MySQL*

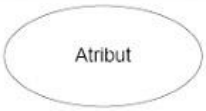
Perangkat lunak *MySQL* adalah perangkat lunak basis data *server* yang terkenal dan bersifat *open-source* dengan dukungan *driver* yang luas dari berbagai *vendor*. *MySQL* adalah sebuah implementasi dari sistem manajemen basis data relasional (*RDBMS*) yang didistribusikan secara gratis dibawah *lisensi GPL* (*General Public License*). Setiap pengguna dapat secara bebas menggunakan *MySQL*, namun dengan batasan perangkat lunak tersebut tidak boleh dijadikan produk turunan yang bersifat komersial. *MySQL* sebenarnya merupakan turunan salah satu konsep utama dalam basis data yang telah ada sebelumnya. *SQL* (*Structured Query Language*). *SQL* adalah sebuah konsep pengoperasian basis data, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis.

Kehandalan suatu sistem basis data (*DBMS*) dapat diketahui dari cara kerja pengoptimasi-nya dalam melakukan proses perintah-perintah *SQL* yang dibuat oleh pengguna maupun program-program aplikasi yang memanfaatkannya. Sebagai peladen basis data, *MySQL* mendukung operasi basis data transaksional maupun operasi basis data *non-transaksional*. Pada modus operasi *non-transaksional*, *MySQL* dapat dikatakan unggul dalam hal unjuk kerja dibandingkan perangkat lunak pengelola basis data *kompetitor* lainnya. Namun demikian pada modus *non-transaksional* tidak ada jaminan atas reliabilitas terhadap data yang tersimpan, karenanya modus *non-transaksional* hanya cocok untuk jenis aplikasi yang tidak membutuhkan reliabilitas data seperti aplikasi *blogging* berbasis *web*, *CMS*, dan sejenisnya. Untuk kebutuhan sistem yang ditujukan untuk bisnis sangat disarankan untuk menggunakan modus basis data *transaksional*, hanya saja sebagai konsekuensinya unjuk kerja *MySQL* pada modus *transaksional* tidak secepat unjuk kerja pada modus *non-transaksional* (Supardi ; 2007: 97).

II.11. *Entity Relationship Diagram (ERD)*

Entity Relationship Diagram atau *ERD* merupakan salah satu alat (*tool*) berbentuk grafis yang populer untuk *desain database. Tool* ini relatif lebih mudah dibandingkan dengan Normalisasi. Kebanyakan sistem analis memakai alat ini, tetapi yang jadi masalah, kalau di cermati secara seksama, *tool* ini mencapai *2NF* (Yuniar Supardi, 2010 : 448).

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 : Yuniar Supardi ; 2010 : 448)

II.12. Kamus Data

Kamus data (data *dictionary*) mencakup definisi-definisi dari data yang disimpan di dalam basis data dan dikendalikan oleh sistem manajemen basis data. Figur 6.5 menunjukkan hanya satu tabel dalam basis data jadwal. 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 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 mempergunakan data tidak akan ikut terpengaruh (Raymond McLeod ; 2008 : 171).

II.13. 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.13.1.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 (*2nd 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 nonkey* 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*).

II.14. Unified Modeling Language (UML)

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.

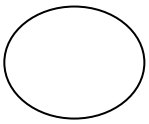
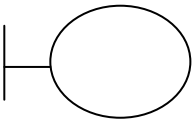
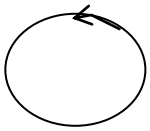
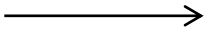
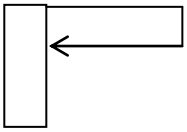
Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan *UML* adalah sebagai berikut :

- 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 :

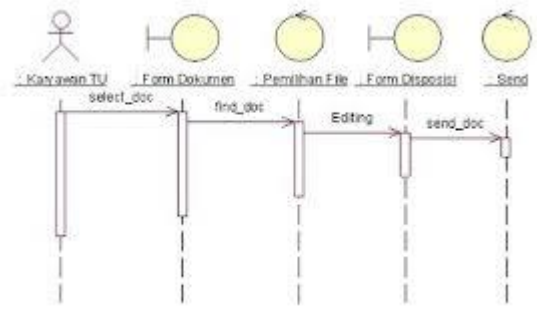
Tabel II.2. Simbol *Sequence Diagram*

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.6 berikut :



Gambar. II.3. *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 tanggung jawab *entitas* yang menentukan perilaku sistem.

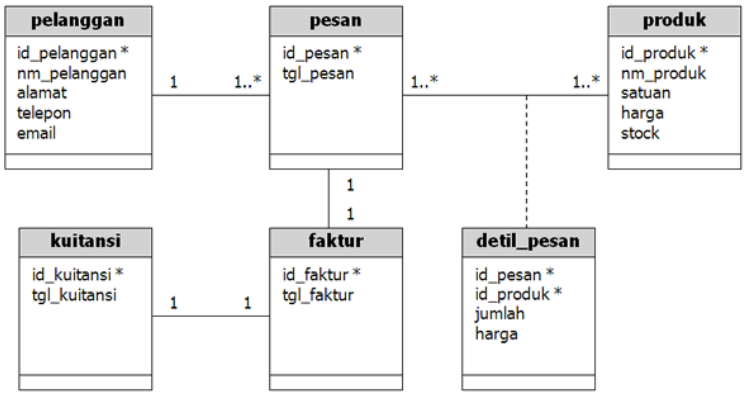
Class diagram juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan. *Class diagram* secara khas meliputi: Kelas (*Class*), Relasi, *Associations*, *Generalization* dan *Aggregation*, Atribut (*Attributes*), Operasi (*Operations/Method*),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

Tabel III.3. *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.7 berikut :

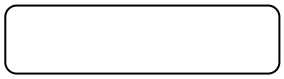
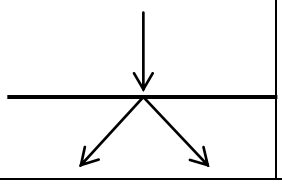


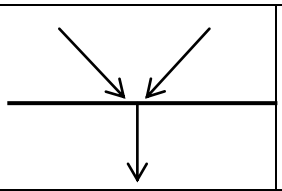
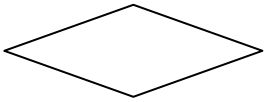
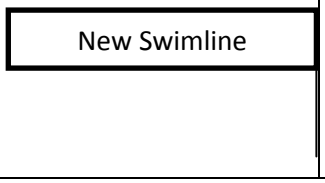
Gambar. II.4. *Class Diagaram*
(Sumber : Windu Gata ; 2013 : 8)

- 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 :

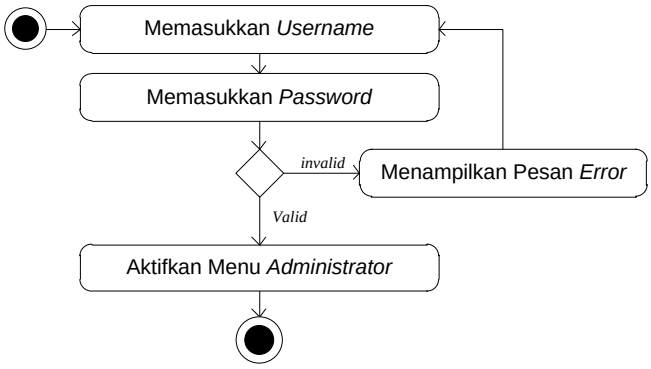
Tabel III.4. Simbol *Activity Diagram*

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.5 berikut :



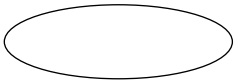
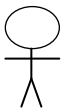
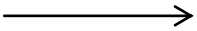
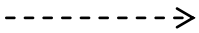
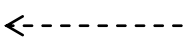
Gambar. II.5. *Activity Diagram*
(Sumber : Windu Gata ; 2013 : 6)

- *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

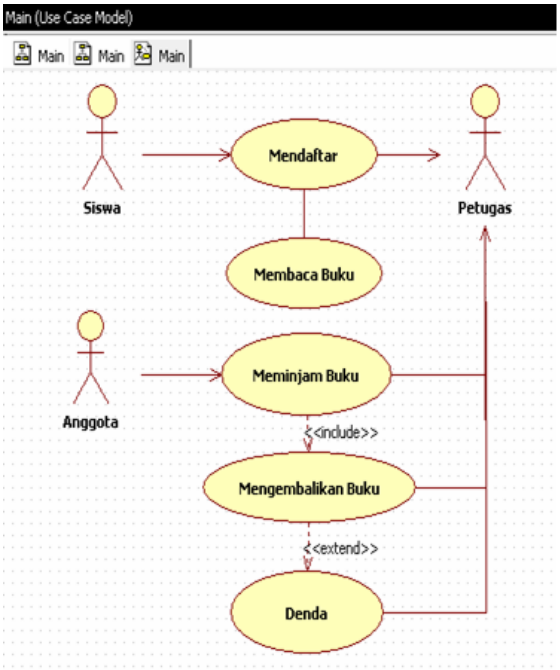
informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbol-simbol yang digunakan dalam *use case* diagram, yaitu :

Tabel III.5. Simbol *Use Case*

Gambar	Keterangan
	*Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar 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.4 berikut :



Gambar. II.6. *Use Case Diagram*
(Sumber : Windu Gata ; 2013 : 4)