BAB II

TINJAUAN PUSTAKA

II.1. Sistem Informasi

II.1.1. Sistem

Sistem merupakan kumpulan elemen yang saling berkaitan yang bertanggung jawab memproses masukan (*input*) sehingga menghasilkan keluaran (*output*) (Kusrini; 2007: 11).

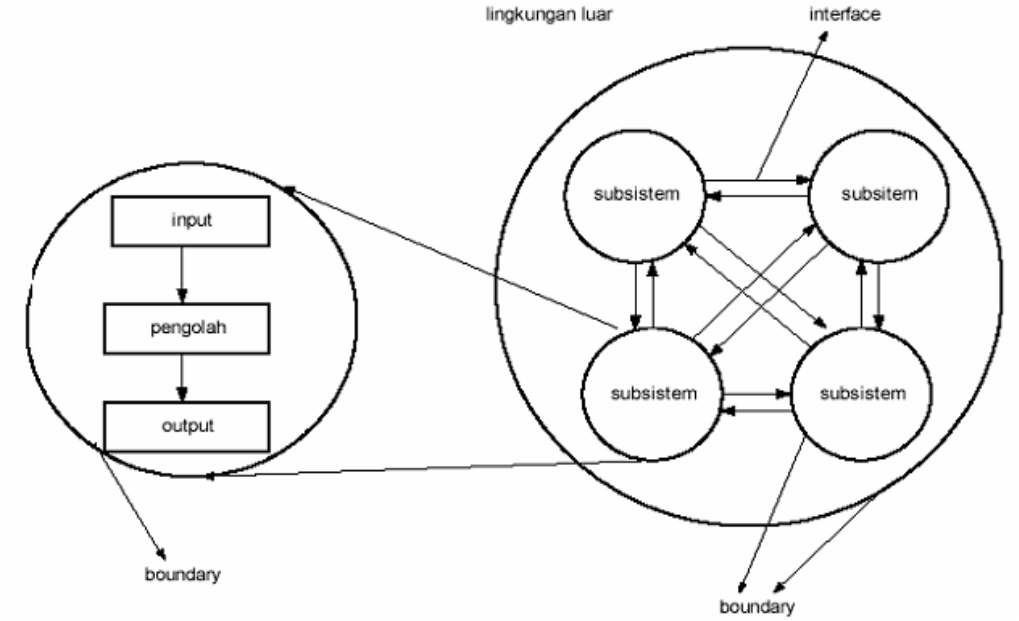
Sistem adalah sebuah tatanan yang terdiri atas sejumlah komponen fungsional (dengan tugas/fungsi khusus) yang saling berhubungan dan secara bersama-sama bertujuan untuk memenuhi suatu proses/pekerjaan tertentu. Sebagai contoh, sistem kendaraan terdiri dari: komponen starter, komponen pengapian, komponen penggerak, komponen pengerem, komponen kelistrikan-spedometer, lampau dan lain-lain. Komponen-komponen tersebut diatas memiliki tujuan yang sama yaitu untuk membuat kendaraan tersebut bisa dikendarai dengan nyaman dan aman. Contoh lain yaitu sistem perguruan tinggi, yang terdiri dari dosen, mahasiswa, kurikulum, dan lain-lain. Sistem ini bertujuan untuk menghasilkan mahasiswa-mahasiswa yang memiliki kemampuan di bidang ilmunya. (Kusrini; 2008: 4)

II.1.2. Konsep Dasar Sistem

Sistem adalah kumpulan dari elemen-elemen yang berinteraksi untuk mencapai suatu tujuan tertentu. Menurut Jerry FithGerald ; sistem adalah suatu

jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau menyelesaikan suatu sasaran tertentu. (Kusrini; 2008: 4)

Berikut ini pada gambar II.1 adalah contoh dari bentuk karakteristik sistem:



Gambar II.1. Karakteristik Sistem
(Sumber : Kusrini; 2008: 5)

1. Komponen Sistem (*Components*)

Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang artinya saling bekerja sama membentuk satu kesatuan. Komponen-komponen sistem atau elemen-elemen sistem dapat berupa suatu subsistem atau bagian-bagian dari sistem. Setiap sistem tidak perduli betapapun kecilnya, selalu mengandung komponen-komponen atau subsistem-subsistem. Setiap subsistem mempunyai sifat-sifat dari sistem untuk menjalankan suatu fungsi tertentu dan mempengaruhi proses sistem secara keseluruhan. Jadi, dapat dibayangkan jika dalam suatu sistem ada

subsistem yang tidak berjalan / berfungsi sebagaimana mestinya. Tentunya sistem tersebut tidak akan berjalan mulus atau mungkin juga sistem tersebut rusak sehingga dengan sendirinya tujuan sistem tersebut tidak tercapai.

2. Batas Sistem (*Boundary*)

Batas sistem (*boundary*) merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainnya atau dengan lingkungan luarnya. Batas sistem ini memungkinkan suatu sistem dipandang sebagai satu kesatuan. Batas suatu sistem menunjukkan ruang lingkup (*scope*) dari sistem tersebut.

3. Lingkungan Luar Sistem (*Environments*)

Lingkungan luar dari suatu sistem adalah apapun diluar batas dari sistem yang mempengaruhi operasi sistem. Lingkungan luar sistem dapat bersifat menguntungkan dan dapat juga bersifat merugikan sistem tersebut. Lingkungan luar yang menguntungkan merupakan energi dari sistem dan dengan demikian harus tetap dijaga dan dipelihara. Sedang lingkungan luar yang merugikan harus ditahan dan dikendalikan, kalau tidak maka akan mengganggu kelangsungan hidup dari sistem.

4. Penghubung (*Interface*)

Penghubung merupakan media penghubung antara satu subsistem dengan subsistem lainnya. Melalui penghubung ini memungkinkan sumber-sumber daya mengalir dari satu subsistem ke yang lainnya. Keluaran (*output*) dari satu subsistem akan menjadi masukan (*input*) untuk

subsistem lainnya dengan melalui penghubung. Dengan penghubung satu subsistem dapat berintegrasi dengan subsistem yang lainnya membentuk satu kesatuan.

5. Masukan (*Input*)

Masukan adalah energi yang dimasukkan ke dalam sistem. Masukan dapat berupa masukan perawatan (*maintenance input*) dan masukan sinyal (*signal input*). Maintenance input adalah energi yang dimasukkan supaya sistem tersebut dapat beroperasi. Signal input adalah energi yang diproses untuk didapatkan keluaran. Sebagai contoh didalam sistem komputer, program adalah maintenance input yang digunakan untuk mengoperasikan komputernya dan data adalah signal input untuk diolah menjadi informasi.

6. Keluaran (*Output*)

Keluaran adalah hasil dari energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna dan sisa pembuangan. Keluaran dapat merupakan masukan untuk subsistem yang lain atau kepada super sistem. Misalnya untuk sistem komputer, panas yang dihasilkan adalah keluaran yang tidak berguna dan merupakan hasil sisa pembuangan, sedang informasi adalah keluaran yang dibutuhkan.

7. Pengolah (*Process*)

Suatu sistem dapat mempunyai suatu bagian pengolah yang akan merubah masukan menjadi keluaran. Suatu sistem produksi akan mengolah masukan berupa bahan baku dan bahan-bahan yang lain menjadi keluaran berupa barang jadi. Sistem akuntansi akan mengolah data-data transaksi

menjadi laporan-laporan keuangan dan laporan-laporan lain yang dibutuhkan oleh manajemen.

8. Sasaran (*Objectives*) atau Tujuan (*Goal*)

Suatu sistem pasti mempunyai tujuan atau sasaran. Kalau suatu sistem tidak mempunyai sasaran, maka operasi sistem tidak akan ada gunanya. Sasaran dari sistem sangat menentukan sekali masukan yang dibutuhkan sistem dan keluaran yang akan dihasilkan sistem. Suatu sistem dikatakan berhasil bila mengenai sasaran atau tujuannya. Perbedaan suatu sasaran (*objectives*) dan suatu tujuan (*goal*) adalah, goal biasanya dihubungkan dengan ruang lingkup yang lebih luas dan sasaran dalam ruang lingkup yang lebih sempit. Bila merupakan suatu sistem utama, seperti misalnya sistem bisnis perusahaan, maka istilah goal lebih tepat diterapkan. Untuk sistem akuntansi atau sistem-sistem lainnya yang merupakan bagian atau subsistem dari sistem bisnis, maka istilah objectives yang lebih tepat. Jadi tergantung dari ruang lingkup mana memandang sistem tersebut. Seringkali tujuan (*goal*) dan sasaran (*objectives*) digunakan bergantian dan tidak dibedakan. (Kusrini; 2008: 8)

II.1.3. Informasi

Informasi adalah data yang dapat dianalogikan dengan data – data , yang belum di kelolah dan harus diolah untuk menjadi informasi yang akurat. Agar informasi yang penulis sajikan lebih bermanfaat maka terlebih dahulu dibuat aliran informasi yang lebih jelas dan lengkap. Berkaitannya dengan penyedia

informasi bagi manajemen dalam mengambil suatu keputusan, yang diperoleh harus berkualitas, maka kualitas dari informasi tergantung :

1. Akurat

   Akurat berarti bahwa informasi harus bebas dari kesalahan - kesalahan dan tidak biasa (menyesatkan) dan jelas mencerminkan maksudnya. Informasi harus akurat karena dari sumber informasi sampai ke penerimaan informasi kemungkinan banyak terjadi gangguan yang dapat merubah informasi atau merusak informasi tersebut.

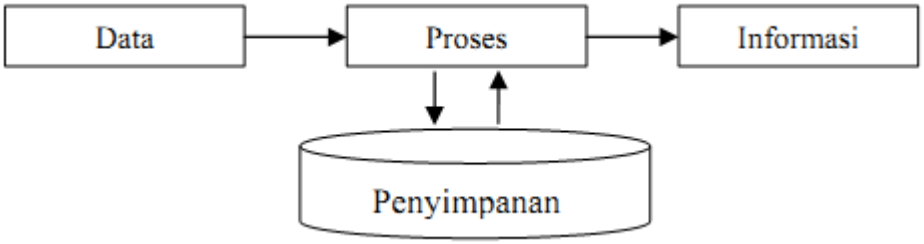
2. Relevansi

   Relevansi berarti bahwa informasi benar – benar berguna bagi suatu tindakan dan keputusan oleh seseorang

3. Tepat waktu

   Tepat waktu berarti bahwa informasi yang datang pada penerimaan tidak boleh terlambat. Informasi yang sudah usang tidak akan mempunyai nilai lagi. Karena informasi merupakan landasan dalam pengambilan keputusan. Dewasa ini mahalnya nilai informasi disebabkan harus cepatnya informasi itu di dapat, Untuk lebih jelasnya informasi merupakan hasil atau output dari proses informasi data (Kusrini : 2008: 15) .

   Proses data menjadi informasi bisa kita lihat pada gambar II.2 berikut ini :



Gambar II.2. Proses Data Menjadi Informasi
(Sumber : Kusrini; 2008: 15)

II.2. Sistem Pendukung Keputusan

Sistem Pendukung Keputusan adalah sistem informasi interaktif yang menyediakan informasi, pemodelan, dan pemanipulasian data. Sistem ini digunakan untuk membantu pengambilan keputusan dalam situasi yang semiterstruktur dan situasi yang tidak terstruktur, dimana tidak seorang pun tahu secara pasti bagaimana keputusan seharusnya dibuat (Kusrini; 2007: 15-16).

DSS (*Decision Support System*) atau Sistem Pendukung Keputusan Sistem Pendukung Keputusan biasanya dibangun untuk mendukung solusi suatu masalah atau untuk mengevaluasi suatu peluang, hal tersebut disebut dengan aplikasi DSS. Aplikasi DSS digunakan dalam pengambilan keputusan.

Aplikasi DSS menggunakan data, memberikan antarmuka pengguna yang mudah dan dapat menggabungkan pemikiran pengambil keputusan. DSS lebih ditujukan untuk mendukung manajemen dalam melakukan pekerjaan yang bersifat analitis dalam situasi yang kurang terstruktur dan dengan kriteria yang kurang jelas (Kurini; 2007: 16).

Secara umum, sistem pendukung keputusan mencakup beberapa komponen utama yaitu DBMS, MBMS dan antarmuka pengguna. Adapun juga komponen opsional yang disebut subsistem manajemen berbasis pengetahuan. Komponen-komponen tersebut dapat dijelaskan sebagai berikut:

1. Manajemen Data (DBMS)

   Subsistem manajemen data memasukkan satu database yang berisi data yang relevan untuk situasi dan dikelola oleh perangkat lunak yang disebut sistem manajemen database (DBMS). Subsistem ini bisa diinterkoneksikan dengan data *warehouse* perusahaan, suatu repositori untuk data perusahaan yang relevan dengan pengambilan keputusan (Kusrini; 2007: 25).

2. Manajemen Model (MBMS)

   Merupakan paket perangkat lunak yang memasukkan model keuangan, statistik, ilmu manajemen, atau model kuantitatif lain yang memberika kapabilitas analitik dan manajemen perangkat lunak yang tepat. Bahasa-bahasa pemodelan untuk membangun model-model *custom* juga dimasukkan. Perangkat lunak itu sering disebut sistem manajemen basis model (MBMS). Komponen tersebut bisa dikoneksikan ke penyimpanan korporat atau eksternal yang ada pada model (Kusrini; 2007: 25).

3. Antarmuka Pengguna

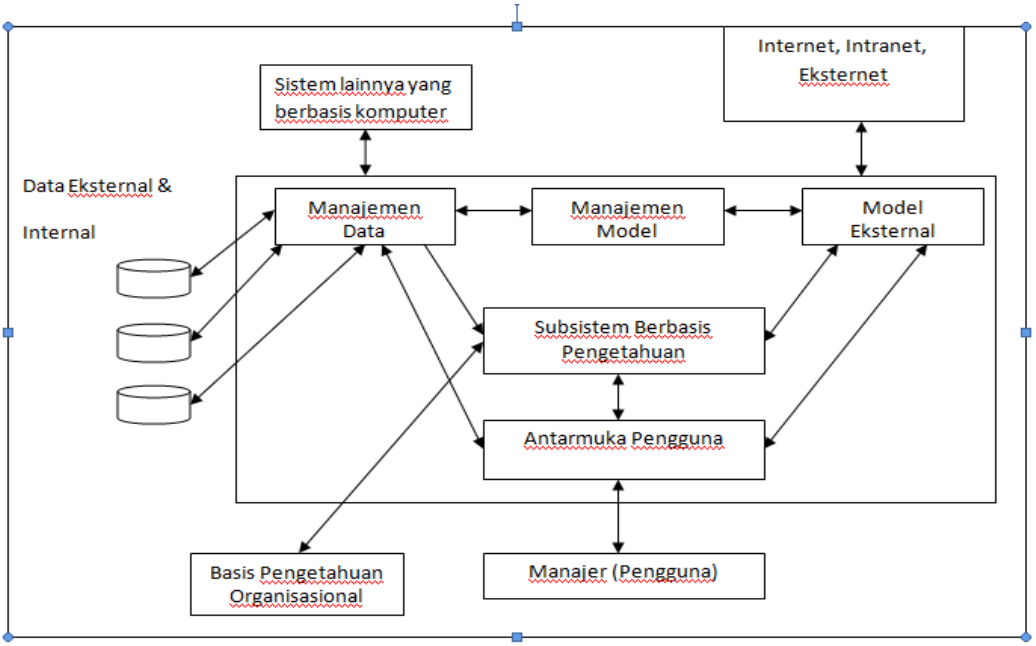
   Pengguna berkomunikasi dengan dan memerintahkan sistem pendukung keputusan melalui subsistem tersebut. Pengguna adalah bagian

yang dipertimbangkan dari sistem. Para peneliti menegaskan bahwa beberapa kontribusi unik dari sistem pendukung keputusan berasal dari interaksi yang intensif antara computer dan pembuat keputusan (Kusrini; 2007: 25).

4. Manajemen Berbasis Pengetahuan

Subsistem tersebut mendukung semua subsistem lain atau bertindak langsung sebagai suatu komponen independen dan bersifat opsional. Selain memberikan intelegensi untuk memperbesar pengetahuan si pengambil keputusan, subsistem tersebut bisa diinterkoneksikan dengan repository pengetahuan perusahaan (bagian dari sistem manajemen pengetahuan) kadang-kadang disebut basis pengetahuan organisasional (Kusrini; 2007: 26).

Arsitektur DSS bisa dilihat pada gambar II.3 berikut ini :



Gambar II.3. Arsitektur DSS
(Sumber : Kusrini; 2007: 26)

II.3 Simple Multi Atribute Rating *(SMART)*

SMART merupakan salah satu metode pengambilan keputusan multi kriteria yang dikembangkan oleh Edward pada tahun 1976. Teknik pengambilan keputusan multi kriteria ini didasarkan pada teori bahwa setiap alternatif terdiri dari sejumlah kriteria yang memiliki nilai-nilai dan setiap kriteria memiliki bobot yang menggambarkan seberapa penting ia dibandingkan dengan kriteria lain. Pembobotan ini digunakan untuk menilai setiap alternatif agar dapat diperoleh alternatif yang terbaik.

SMART merupakan metode pengambilan keputusan yang fleksibel untuk meramal nilai dari setiap alternatif. *SMART* banyak digunakan karena kesederhanaannya dalam merespon kebutuhan pembuat keputusan dan caranya menganalisa respon tersebut. Analisa yang terlibat bersifat transparan sehingga metode ini memberikan pemahaman masalah yang tinggi dan dapat diterima oleh pembuat keputusan (Jurnal EECCIS; 2014: Vol. 8 No. 1).

Ada beberpa tahapan untuk menyeleseikan suatu kasus menggunakan metode *SMART* yaitu :

1. Langkah 1 : menentukan jumlah kriteria
2. Langkah 2 : sistem secara default memberikan skala 0-100 berdasarkan prioritas yang telah diinputkan kemudian dilakukan normalisasi :

$$\text{Normalisasi} = \frac{w_j}{\sum w_j}$$

Keterangan :

w_j = bobot suatu kriteria

$\sum w_j$ = total bobot semua kriteria

3. Langkah 3 : memberikan nilai kriteria untuk setiap alternatif.
4. Langkah 4 : hitung nilai utility untuk setiap kriteria masing-masing.

$$u_i(a_i) = 100\frac{(C_{\max} - C_{outi})}{(C_{\max} - C_{\min})}\%$$

Keterangan :

$u_i(a_i)$: nilai utility kriteria ke-1 untuk kriteria ke-i

C_{max} : nilai kriteria maksimal

C_{min} : nilai kriteria minimal

$C_{out\ i}$: nilai kriteria ke-i

5. Langkah 5 : hitung nilai akhir masing-masing :

$$u(a_i) = \sum_{J=1}^{m} w_j u_i(a_i),$$

Keterangan:

w_j = nilai pembobotan kriteria ke-j dan *k* kriteria

$u(a_i)$ = nilai utility kriteria ke-i untuk kriteria ke-i

Pemilihan keputusan adalah mengidentifikasi mana dari n alternatif yang mempunyai nilai fungsi terbesar.

II.4. Microsoft SQL Server 2008

SQL (*Structured Query Language*) adalah sebuah bahasa yang dipergunakan untuk mengakses data dalam basis data relasional. Bahasa ini secara *de facto* merupakan bahasa standar yang digunakan dalam manajemen basis data relasional. Saat ini hampir semua *server* basis data yang ada mendukung bahasa ini untuk melakukan manajemen datanya. (Jurnal Sistem Informasi, Vol. 6 No. 2; Adelia, Jimmy Setiawan ; 2011 : 115)

SQL Server 2008 adalah sebuah terobosan baru dari Microsoft dalam bidang *database* dibuat pada saat kemajuan dalam bidang *hardware* sedemikian pesat. SQL Server adalah DBMS (Database Management *System*) yang dibuat oleh Microsoft untuk ikut berkecimpung dalam persaingan dunia pengolahan data menyusul pendahulunya seperti IBM dan Oracle.

II.5. UML (*Unified Modelling Language*)

Unified Modeling Language (UML) adalah suatu alat untuk memvisualisasikan dan mendokumentasikan hasil analisa dan desain yang berisi sintak dalam memodelkan sistem secara visual (Braun, *et. al*.2011). Juga merupakan satu kumpulan konvensi pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem *software* yang terkait dengan objek (Whitten, *et. al*. 2004)

UML merupakan salah satu alat bantu yang sangat handal dalam bidang pengembangan sistem berorientasi objek karena UML menyediakan bahasa

pemodelan visual yang memungkinkan pengembang sistem membuat *blue print* atas visinya dalam bentuk yang baku. UML berfungsi sebagai jembatan dalam berkomunikasikan beberapa aspek dalam sistem melalui sejumlah elemen grafis yang bisa dikombinasikan menjadi diagram. UML mempunyai banyak diagram yang dapat mengakomodasi berbagai sudut pandang dari suatu perangkat lunak yang akan dibangun.(Yuni Sugiarti; 2013:36-37)

Dengan menggunakan UML, kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemograman apapun. Tetapi karena UML juga menggunakan dan operation dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C.

Seperti bahasa-bahasa lainnya, UML mengidentifikasikan notasi dan syntax / semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML syntax mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Software Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*) (Yuni Sugiarti;2013:34).

Blok pembangunan utama UML adalah diagram. Beberapa diagram ada yang rinci (jenis *timming diagram*) dan lainnya ada yang bersifat umum (misalnya diagram kelas). Para pengembang sistem berorientasi objek menggunakan bahasa model untuk menggambarkan, membangun dan mendokumentasikan sistem yang mereka rancang. UML memungkinkan para anggota team untuk bekerja sama dengan bahasa model yang sama dengan mengaplikasikan beragam sistem. Intinya UML merupakan alat komunikasi yang konsisten dalam mendukung para pengembang sistem saat ini.

1. *Use Case Diagram*

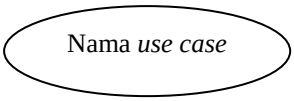
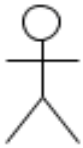
Dalam membuat sebuah sistem, langkah awal yang perlu dilakukan adalah menentukan kebutuhan. Terdapat dua jenis kebutuhan, yaitu kebutuhan fungsional dan kebutuhan nonfungsional. Kebutuhan fungsional adalah kebutuhan pengguna dan stakeholder sehari-hari yang akan dimiliki oleh sistem, dimana kebutuhan ini akan digunakan oleh pengguna *stakeholder*. Sedangkan kebutuhan nonfungsional adalah kebutuhan yang memperhatikan hal-hal berikut yaitu performansi, kemudahan dalam menggunakan sistem, kehandalan sistem, keamanan sistem, keuangan, legalitas, dan operasional.

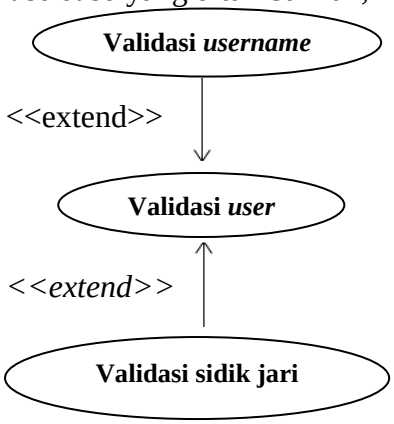
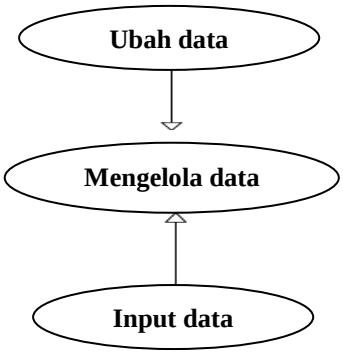
Kebutuhan fungsi akan digambarkan melalui sebuah diagram yang dinamakan diagram use case. *Use case* diagram atau diagram use case merupakan pemodelan untuk menggambarkan kelakuan (*behavior*) sistem yang akan dibuat. Diagram *use case* mendeskripsikan sebuah interaksi antar satu atau lebih actor dengan sistem yang akan dibuat. Dengan pengertian yang cepat, diagram use case digunakan untuk mengetahui fungsi apa saja yang ada didalam sebuah sistem dan

siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Terdapat beberapa simbol dalam menggambarkan diagram use case, yaitu use case, actor dan relasi. Hal perlu diingat mengenai diagram use case adalah diagram use case bukan menggambarkan tampilan antarmuka (*user interface*), arsitektur dari sistem, kebutuhan nonfungsional, dan tujuan performansi. Sedangkan untuk penamaan *use case* adalah nama didefinisikan sesimpel mungkin, dapat dipahami dan menggunakan kata kerja (Yuni Sugiarti; 2013:41).

Berikut ini pada tabel II.1. adalah simbol-simbol yang ada pada diagram use case:

Tabel II.1. Simbol – Simbol *Use Case* Diagram

Simbol	Deskripsi
Use case 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, biasanya dinyatakan dengan menggunakan kata kerja diawal - awal frase nama *use case*
Aktor / *actor*  **nama aktor**	orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari *actor* adalah gambar orang, tapi *aktor* belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal nama *aktor*.
Asosiasi/ *association* _____	Komunikasi antara *actor* dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan actor

Ekstensi / *extend* <<*extend*>> 	Relasi *use case* tambahan kesebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu: mirip dengan prinsip *inheritance* pada pemograman berorientasi objek; biasanya *use*
	case tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan, misal  Arah panah mengarah pada *use case* yang ditambahkan; biasanya *use case* yang menjadi *extend*-nya merupakan jenis yang sama dengan *use case* yang menjadi induknya.
Generalisasi/*generalization* 	Hubungan generalisasi dan spesialisasi (umum – khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya, misalnya:  Arah panah mengarah pada use case yang menjadi generalisasinya (umum).

Menggunakan / *include* / *uses* <<*include*>>  <<*uses*>> 	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat *use case* ini. Ada dua sudut pandang yang cukup besar mengenai *include* di *use case* : *include* berarti *use case* yang ditambahkan akan selalu dipanggil saat *use case* tambahan dijalankan. *include* berarti *use case* yang tambahan akan selalu melakukan pengecekan apakah *use case* yang ditambahkan telah dijalankan sebelum *use case* tambahan dijalankan.

Sumber : (Rosa A.S, M.Shalahuddin; 2013; 155-158).

2. Diagram Kelas (*Class Diagram*)

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. kelas memiliki apa yang disebut atribut dan metode atau operasi.

1. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas.
2. Atribut mendeskripsikan property dengan sebaris teks didalam kotak kelas tersebut.
3. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

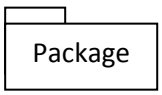
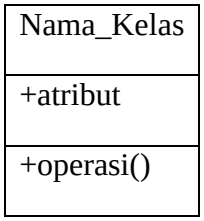
Diagram kelas mendeskripsikan jenis-jenis objek dalam sistem dan berbagai hubungan statis yang terdapat diantara mereka. Diagram kelas juga menunjukkan properti dan operasi sebuah kelas dan batasan-batasan yang terdapat dalam hubungan-hubungan objek. Diagram kelas menggambarkan struktur dan deskripsi class, package dan objek beserta hubungan satu sama lain seperti containment, perwarisan, asosiasi, dan lain-lain (Yuni Sugiarti;2013:57).

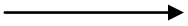
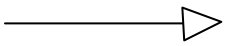
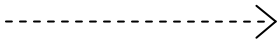
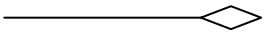
Kelas memiliki tiga area pokok :

1. Nama
2. Atribut
3. Operasi

Berikut ini pada table II.2 adalah simbol-simbol pada diagram kelas :

Tabel II.2. Simbol-Simbol Diagram Kelas

Simbol	**Deskripsi**
Package 	Package merupakan bungkusan dari satu kelas atau lebih
Operasi 	Kelas pada struktur system
Antarmuka / *interface*  Nama *interface*	Sama dengan konsep *interface* dalam pemrograman berorientasi objek

Asosiasi / *association* 1 1..*	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*
Asosiasi berarah / *Directed association* 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*
Generalisasi 	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus)
Kebergantungan 	Relasi antar kelas dengan makna Kebergantungan antar kelas
Agregasi / *aggregation* 	Relasi antar kelas dengan makna Semua bagian (*whole part*)

Sumber : (Yuni Sugiarti;2013:59).

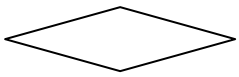
3. *Activity diagram*

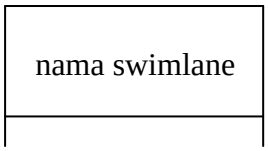
Diagram aktivitas atau *activity diagram* menggambarkan *workflow*(aliran kerja) atau aktivias dari sebuah sistem atau proses bisnis. *Activity diagram* menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, decision yang mungkin terjadi, dan bagaimana mereka berakhir.

Activity Diagran merupakan state diagram khusus, dimana sebagian besar state adalah action dan sebagian besar transisi di-*trigger* oleh selesainya state sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tdak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarka proses- proses dan jalur-jalur aktivitas dari level atas secara umum (Yuni Sugiarti;2013:75).

Berikut adalah tabel II.3. yang menggambarkan simbol-simbol yang digunakan pada diagram aktivitas :

Tabel II.3. Simbol *Activity Diagram*

Simbol	Deskripsi
Status awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
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.

Status akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.
Swimlane 	Memisahkan organisasi bisnis yang bertanggungjawab terhadap aktivitas yang terjadi.

Sumber : (Rosa A.S, M.Shalahuddin; 2013; 161-163)

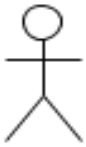
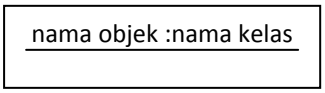
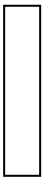
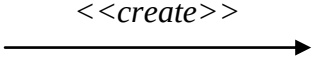
4. *Squence Diagram*

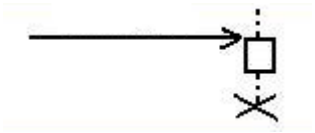
Diagram sekuen menggambarkan kelakuan/perilaku objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek.

Banyaknya diagram Sekuen yang digambarkan adalah sebanyak pendefenisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefenisikan interaksi jalanya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *use case* didefenisikan maka diagram sekuen yang harus dibuat juga semakin banyak (Yuni Sugiarti;2013:69).

Berikut adalah tabel II.6. yang menerangkan simbol-simbol yang ada pada diagram sekuen:

Tabel II.6. Diagram Sequence

Simbol	Deskripsi
Aktor  **nama aktor**	orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari *actor* adalah gambar orang, tapi *actor* belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal nama *actor*.
Garis hidup / *lifeline*	Menyatakan kehidupan suatu objek.
Objek 	Menyatakan objek yang berinteraksi pesan.
Waktu aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah tahapan yang dilakukan didalamnya.
Pesan tipe create 	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat

Pesan tipe call 1 : nama_metode{} 	Menyatakan suatu objek memanggil operasi / metode yang ada pada objek lain atau dirinya sendiri.
Pesan tipe send 1 : masukan 	Menyatakan bahwa suatu objek mengirimkan data / masukan / informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi.
Pesan tipe return 1 : keluaran 	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian.
Pesan tipe destroy <<*destroy*>> 	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada creater maka ada *destroy*.

Sumber : (Rosa A.S, M.Shalahuddin; 2013; 165-167)

II.6. Visual Basic 2010

Microsoft Visual Basic 2010 (sering disingkat sebagai *VB* saja) merupakan sebuah bahasa pemrograman yang bersifat *event driven* dan menawarkan *Integrated Development Environment (IDE)* visual untuk membuat program aplikasi berbasis sistem operasi *Microsoft Windows* dengan menggunakan model pemrograman *Common Object Model (COM). Visual Basic* merupakan turunan bahasa *BASIC* dan menawarkan pengembangan aplikasi komputer berbasis grafik dengan cepat, akses ke basis data menggunakan Data Access Objects (DAO), Remote Data Objects (RDO), atau ActiveX Data Object (ADO), serta menawarkan pembuatan kontrol ActiveX dan objek ActiveX.

Visual Basic merupakan turunan bahasa *BASIC* dan menawarkan pengembangan aplikasi komputer berbasis grafik dengan cepat, akses ke basis data menggunakan Data Access Objects (DAO), Remote Data Objects (RDO), atau ActiveX Data Object (ADO), serta menawarkan pembuatan kontrol ActiveX dan objek ActiveX. Beberapa bahasa skrip seperti Visual Basic for Applications (VBA) dan Visual Basic Scripting Edition (VBScript), mirip seperti halnya Visual Basic, tetapi cara kerjanya yang berbeda. (Jurnal Sistem Informasi, Vol. 6 No. 2; Adelia, Jimmy Setiawan ; 2011 : 114-115).