BAB II

LANDASAN TEORI

II.1. Sistem Pendukung Keputusan

Konsep Sistem Pendukung Keputusan pertama kali diperkenalkan pada awal tahun 1970-an oleh Michael S. Scott Morton dengan istilah *Management Decision System* (Sprague, 1982). Konsep pendukung keputusan ditandai dengan sistem interaktif berbasis komputer yang membantu pengambil keputusan memanfaatkan data dan model untuk menyelesaikan masalah-masalah yang tidak terstruktur.

Pada dasarnya SPK dirancang untuk mendukung seluruh tahap pengambilan keputusan mulai dari mengidentifikasi masalah, memilih data yang relevan, menentukan pendekatan yang digunakan dalam proses pengambilan keputusan, sampai mengevaluasi pemilihan alternatif. Menurut Simon (Suryadi dan Ramdhani, model yang menggambarkan proses pengambilan keputusan. Proses ini terdiri dari tiga fase, yaitu sebagai berikut.

a. *Intelligence*

Tahap ini merupakan proses penelusuran dan pendeteksian dari lingkup problematika serta proses pengenalan masalah. Data masukan diperoleh, diproses, dan diuji dalam rangka mengindentifikasi masalah.

b. *Design*

Tahap ini merupakan proses menemukan, mengembangkan, dan menganalisis alternatif tindakan yang bisa dilakukan. Tahap ini meliputi

proses untuk mengerti masalah, menurunkan solusi dan menguji kelayakan solusi.

c. *Choice*

Pada tahap ini dilakukan proses pemilihan diantara berbagai alternatif tindakan yang mungkin dijalankan. Hasil pemilihan tersebut kemudian diimplementasikan dalam proses pengambilan keputusan.

Meskipun implementasi termasuk tahap ketiga, namun ada beberapa pihak berpendapat bahwa tahap ini perlu dipandang sebagai bagian yang terpisah guna menggambarkan hubungan antar fase secara lebih komprehensif.

II.1.1. Jenis – Jenis Keputusan

Keputusan-keputusan yang dibuat pada dasarnya dikelompokkan dalam 2 jenis, antara lain:

1. Keputusan Terperogram

   Keputusan ini bersifat berulang dan rutin, sedemikian hingga suatu prosedur pasti telah dibuat menanganinya sehingga keputusan tersebut tidak perlu diperlakukan *de bovo* (sebagai sesuatu yang baru) tiap kali terjadi.

2. Keputusan Tak Terprogram

   Keputusan ini bersifat baru, tidak tersetruktur jarang konsekuen. Tidak ada metode yang pasti untuk menangani masalah ini karena belum ada sebelumnya atau karena sifat dan struktur persisnya tak terlihat atau rumit atau karena begitu pentingnya sehingga memerlukan perlakuan yang sangat khusus (Yunitarini, 2013, hal. 46).

II.1.2. Karakter Sistem Pendukung Keputusan

SPK, menurut tinjaun konotatif, merupakan system yang ditujukan kepada tingkatan manajemen sebagai berikut : (magdalena, 2012, hal. 50-51).

Berfokus pada keputusan., ditujukan pada manajer puncak dan pengambil keputusan.

1. Menekankan pasa fleksibilitas, adaptabilitas, dan respon yang cepat.
2. Mampu mendukung berbagai gaya pengambilan keputusan dan masing-masing pribadi manajer.

II.1.3. Keterbatasan Sistem Pendukung Keputusan

Keterbatasan sistem pendukung keputusan terbagi atas dua yaitu: (magdalena, 2012, hal. 51).

1. Adanya gambaran bahwa SPK seakan-akan hanya dibutuhkan pada tingkat manajemen puncak. Pada kenyataannya, dukungan bagi pengambilan keputusan dibutuhkan pada semua tingkatan manajemen dalam suatu organisasi.
2. Pengambilan keputusan yang terjadi pada beberapa level harus dikoordinasikan. Jadi, dimensi dan pendukung keputusan adalah komunikasi dan koordinasi diantara pengambil keputusan antar level organisasi yang berbeda maupun pada level organisasi yang sama.

II.2. Metode SMART *(Simple Multi-Atribute Rating Technique)*

Pada hakekatnya *Simple Multi-Atribute Rating Technique* (SMART) merupakan suatu model pengambil keputusan yang komrehensif dengan memperhitungkan hal-hal yang bersifat kualitatif dan kuantatif. Dalam model pengambilan keputusan dengan SMART pada dasarnya berusaha menutupi setiap kekurangan dari model-model tanpa komputerisasi sebelumnya. SMART juga memungkinkan ke struktur suatu sistem dan lingkungan kedalam komponen saling berinteraksi dan kemudian menyatukan mereka dengan mengukur dan mengatur dampak dari komponen kesalahan sistem.

Metode pembobotan SMART merupakan metode pendukung keputusan yang paling sederhana. Dalam metode ini dilihat beberapa parameter yang menjadi penentu keputusan tersebut, Parameter tersebut mempunyai range nilai dan bobot yang berbeda-beda. Nilai tersebut nantinya akan menjadi penentu keputusan yang diambil (Yunianti, 2015, hal. 56).

SMART merupakan metode dalam pengambilan keputusan multiatribut. Teknik pengambilan keputusan multiatribut ini digunakan untuk mendukung pembuat keputusan dalam memilih beberapa alternatif. Setiap pembuat keputusan harus memiliki sebuah alternatif yang sesuai dengan tujuan yang dirumuskan. Setiap alternatif terdiri dari sekumpulan atribut dan setiap atribut mempunyai nilai-nilai. Nilai ini dirata-rata dengan skalan tertentu. Setiap atribut mempunyai bobot yang menggambarkan seberapa penting suatu atribut dibandingkan dengan atribut lain. Pembobotan dan pemberian peringkat ini digunakan untuk menilai setiap alternatif agar diperoleh alternatif terbaik (Yunitarini, 2013, hal. 46).

II.2.1. Perhitungan Metode *SMART*

Adapun langkah-langkah dalam proses perhitungan metode *SMART* dapat ditunjukkan sebagai berikut (Kustiyahningsih & Syafa'ah, 2015, hal. 22-23).

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

   $$\text{Normalisasi} = \frac{w_j}{\sum w_j} \quad (1)$$

   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.

   $$u_i(a_i) = 100\frac{(C_{out\,i} - C_{\min})}{(C_{\max} - C_{\min})}\% \quad (2)$$

   …(4)

   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.

   $$u(a_i) = \sum_{J=1}^{m} w_j u_i(a_i), \quad (3)$$

II.2.2. Kelebihan Metode *SMART*

SMART memiliki beberapa kelebihan dibandingkan dengan metode pengambilan keputusan yang lain yaitu: (Yunitarini, 2013, hal. 46-47)

1. Mungkin melakukan penambahan atau pengurangan alternatif pada metode *SMART* penambahan atau pengurangan alternatif tidak akan mempengaruhi perhitungan pembobotan karena setiap penilaian alternatif tidak saling bergantung.
2. Sederhana

   Perhitungsn pada metode *SMART* lebih sederhana sehingga tidak diperlukan matematis yang rumit dengan pemahaman matematika yang kuat.
3. Transparan

   Proses dalam menganalisa alternatif dan kriteria dalam *SMART* dapat dilihat oleh user sehingga user dapat memahami bagaimana alternatif tertentu dapat dipilih. Alasan-alasan bagaimana alternatif itu dipilih dapat dilihat dari prosedur-prosedur yang dilakukan dalam *SMART* mulai dari penentuan kriteria, pembobotan dan pemberian nilai pada setiap alternatif.
4. Fleksibelitas Pembobotan.

   Pembobotan yang dipakai di dalam metode *SMART* ada 3 jenis yaitu pembobotan secara langsung *(direct weigting),* pembobotan swing *(swing weighting),* pembobotan centroid *(centroid weighting).*

II.3. Pengertian Basis Data

Basis data dapat dipahami sebagai suatu kumpulan data terhubung (*interrelated data*) yang disimpan secara bersama–sama pada satu media, tanpa mengatap satu sama lain atau tidak perlu suatu kerangkapan data (Kalaupun ada maka kerangkapan data tersebut harus seminimal mungkin dan terkontrol (*controlled redundancy*), data disimpan dengan cara-cara tertentu sehingga mudah digunakan/atau ditampilkan kembali, data dapat digunakan oleh satu atau lebih program-program aplikasi secara optimal, data disimpan tanpa mengalami ketergantungan dengan program yang akan menggunakannya, data disimpan sedemikian rupa sehingga proses penambahan, pengambilan, dan modifikasi data dapat dilakukan dengan mudah dan terkontrol (Sutanta, 2011, hal. 29-30).

Istilah basis data tentu saja berbeda dengan sistem basis data, istilah sisitem basis data dapat didefenisikan sebagai sekumpulan subsistem yang terdiri atas basis data dengan para pemakai yang menggunakan basis data secara bersama-sama, personal-personal yang merancang dan mengelola basis data, teknik – teknik untuk merancang dan mengelola basis data, serta sistem komputer untuk mendukungnya (Sutanta, 2011, hal. 32-33).

II.3.1. Tujuan Basis Data

Secara lebih lengkap pemanfaatan basis data dilakukan untuk memenuhi tujuan berikut ini (Hidayatullah, 2012, hal. 138).

1. Kecepatan dan kemudahan (*Speed*)
2. Efisiensi ruang penyimpanan (*Space*)
3. Keakuratan (*Accuracy*)

4. Ketersediaan (*Availability*)
5. Kelengkapan (*Completeness*)
6. Keamanan (*Security*)
7. Pemakaian bersama (*Sharability*)

II.4. *Entity Relationship Diagram* (ERD)

Relasi antara dua file atau dua tabel dapat dikategorikan menjadi 3 macam. Demikian pula untuk membantu menggambarkan relasi secara lengkap terdapat juga beberapa relasi dalam hubungan atribut yang ada dalam satu atau dua file.

Adapun relasi yang terjadi di antara dua himpunan entitas dapat berupa sebagai berikut:

1. *One* to *one relationship* dua file

   Hubungan antara file pertama dengan file kedua adalah satu berbanding satu. Seperti pada pelajaran privat di mana satu guru mengajar satu siswa dan satu siswa hanya diajar oleh satu guru.

2. *One* to *many relatioship* dua file

   Hubungan antara file pertama dengan file kedua adalah satu berbanding banyak atau dapat pula dibalik, banyak lawan satu. Seperti pada sistem pengajaran di sekolah dasar, di mana satu guru mengajar banyak siswa dan siswa hanya diajar oleh satu guru.

3. *Many* to *many relationship* dua file

   Hubungan antara file pertama dengan file kedua adalah banyak berbanding banyak. Seperti pada sistem pengajaran di perguruan tinggi, di mana satu

dosen mengajar banyak mahasiswa dan mahasiswa diajar oleh banyak dosen. (Sutabri, 2012, hal. 144).

II.5. Kamus Data

Kamus data adalah katalog fakta tentang data dan kebutuhan informasi suatu sistem informasi. Kamus data terdapat pada tahapan analisis dan perancangan. Pada tahap analisis, kamus data berfungsi untuk mendifinisikan data yang mengalir pada sistem. Sedangkan pada tahap perancangan, kamus data ini digunakan untuk merancang masukan dan keluaran seperti laporan serta basis data. Pada DFD aliran data memiliki sifat global, sedangkan pada kamus data dibuat berdasarkan aliran data yang terdapat pada DFD. (Indrajani, 2015, hal. 30)

Tabel II.1. Notasi Kamus Data

Notasi	**Keterangan**
=	*Is Composed Of*
+	*And*
( )	*Optional (May be present or absent)*
{ }	*Iteration*
[ ]	*Select one of several alternative choices*
**	*Comment*
@	*Identifier (key field) for a store*
|	*Separates alternative choices in the construct*

(Sumber : Indrajani ; 31)

Contoh kamus data, antara lain:

name = courtesy-title + first-name +(middle-name) + last-name

courtesy-title = [Mr. | Miss | Mrs. | Ms. | Dr. | Profesor]

first-name = {legal-character}

middle-name = {legal-character}

last-name = {legal-character}

legal-character = [A-Z|a-z|0-9|'|-| |]

II.6. Normalisasi

Normalisasi adalah suatu teknik dengan pendekatan *bottom-up* yang digunakan untuk membantu mengidentifikasikan hubungan. Dimulai dari menguji hubungan, yaitu *function dependencies* antara atribut. Pengertian lainnya adalah suatu teknik yang menghasilkan sekumpulan hubungan dengan sifat-sifat yang diinginkan dan memenuhi kebutuhan pada perusahaan (Indrajani, 2015, hal. 7).

Proses normalisasi menghasilkan relasi yang optimal, yaitu (Martin, 1975) : (Sutanta, 2011, hal. 175).

1. Memiliki struktur *record* yang konsisten secara logik;
2. Memiliki struktur *record* yang mudah untuk dimengerti;
3. Memiliki struktur *record* yang sederhana dalam pemeliharaan;
4. Memiliki struktur *record* yang mudah ditampilkan kembali untuk memenuhi kebutuhan pengguna;
5. Minimalisasi kerangkapan data guna meningkatkan kinerja sistem.

Secara berturut-turut masing-masing level normal tersebut dibahas berikut ini, dimulai dari bentuk tidak normal. (Sutanta, 2011, hal. 176-179).

1. Relasi bentuk tidak normal (*Un Normalized Form* / UNF)

   Relasi-relasi yang dirancang tanpa mengindahkan batasan dalam defisi basis data dan karakteristik *Relational Database Management System* (RDBM) menghasilkan relasi *Un Normalized Form* (UNF). Bentuk ini harus di hindari dalam perancangan relasi dalam basis data. Relasi *Un Normalized Form* (UNF) mempunyai kriteria sebagai berikut.

   a. Jika relasi mempunyai bentuk *non flat file* (dapat terjadi akibat data disimpan sesuai dengan kedatangannya, tidak memiliki struktur tertentu, terjadi duplikasi atau tidak lengkap)
   b. Jika relasi membuat *set atribut* berulang *(non single values)*
   c. Jika relasi membuat *atribut non atomic value*

2. Relasi bentuk normal pertama (*First Norm Form* / 1NF)

   Relasi disebut juga *First Norm Form* (1NF) jika memenuhi kriteria sebagai berikut.

   a. Jika seluruh atribut dalam relasi bernilai *atomic (atomic value)*
   b. Jika seluruh atribut dalam relasi bernilai tunggal *(single value)*
   c. Jika relasi tidak memuat set atribut berulang
   d. Jika semua record mempunyai sejumlah atribut yang sama.

Permasalahan dalam *First Norm Form* (1NF) adalah sebagai berikut.

   a. Tidak dapat menyisipkan informasi parsial
   b. Terhapusnya informasi ketika menghapus sebuah *record*

3. Bentuk normal kedua *(Second Normal Form* / 2NF)

Relasi disebut sebagai *Second Normal Form (*2NF) jika memenuhi kriteria sebagai berikut

a. Jika memenuhi kriteria *First Norm Form (*1NF)
b. Jika semua atribut nonkunci *Functional Dependence* (FD) pada *Primary Key* (PK)

Permasalahan dalam *Second Normal Form /* 2NF adalah sebagai berikut

a. Kerangkapan data *(data redundancy)*
b. Pembaharuan yang tidak benar dapat menimbulkan inkonsistensi data *(data inconsistency)*
c. Proses pembaharuan data tidak efisien

Kriteria tersebut mengidentifikasikan bahwa antara atribut dalam *Second Normal Form* masih mungkin mengalami *Third Norm Form*. Selain itu, relasi *Second Normal Form* (2NF) menuntut telah didefinisikan atribut *Primary Key* (PK) dalam relasi. Mengubah relasi *First Norm Form* (1NF) menjadi bentuk *Second Normal Form* (2NF) dapat dilakukan dengan mengubah struktur relasi dengan cara :

a. Identifikasikan *Functional Dependence* (FD) relasi *First Norm Form* (1NF)
b. Berdasarkan informasi tersebut, dekomposisi relasi *First Norm Form* (1NF) menjadi relasi-relasi baru sesuai *Functional Dependence* nya. Jika menggunakan diagram maka simpul-simpul yang berada pada puncak diagram ketergantungan data bertindak *Primary Key* (PK) pada relasi baru

4. Bentuk normal ketiga (*Third Norm Form* / 3NF)

Suatu relasi disebut sebagai *Third Norm Form* jika memenuhi kriteria sebagai berikut.

a. Jika memenuhi kriteria *Second Normal Form* (2NF)
b. Jika setiap atribut nonkunci tidak (TDF) *(Non Transitive Dependeny)* terhadap *Primary Key* (PK)

Permasalahan dalam *Third Norm Form* (3NF) adalah keberadaan penentu yang tidak merupakan bagian dari *Primary Key* (PK) menghasilkan duplikasi rinci data pada atribut yang berfungi sebagai *Foreign Key* (FK) (duplikasi berbeda dengan keterangan data).

Mengubah relasi *Second Normal Form* (2NF) menjadi bentuk *Third Norm Form* (3NF) dapat dilakukan dengan mengubah struktur relasi dengan cara :

a. Identifikasi TDF relasi *Second Normal Form* (2NF)
b. Berdasarkan informasi tersebut, dekomposisi relasi *Second Normal Form* (2NF) menjadi relasi-relasi baru sesuai TDF-nya.

5. Bentuk normal *Boyce-Cood (Boyce-Codd Norm Form* / BCNF)

   Bentuk normal *Boyce-Codd Norm Form* (BCNF) dikemukakan oleh R.F. Boyce dan E.F. Codd. Suatu relasi disebut sebagai *Boyce-Codd Norm Form* (BCNF) jika memenuhi kriteria sebagai berikut.

   a. Jika memenuhi kriteria *Third Norm Form* (3NF)
   b. Jika semua atribut penentu (determinan) merupakan CK

6. Bentuk normal keempat (*Forth Norm Form* / 4NF)

   Relasi disebut sebagi *Forth Norm Form* (4NF) jika memenuhi kriteria sebagai berikut.

a. Jika memenuhi kriteria *Boyce-Codd Norm Form*.

b. Jika setiap atribut didalamnya tidak mengalami ketergantungan pada banyak nilai.

7. Bentuk normal kelima (*Fifth Norm Form* / 5NF)

   Suatu relasi memenuhi kriteria *Fifth Norm Form* (5NF) jika kerelasian antar data dalam relasi tersebut tidak dapat direkonstruksi dari struktur relasi yang sederhana.

8. Bentuk normal kunci domain (*Domain Key Norm Form* / DKNF)

   Relasi disebut sebagai *Domain Key Norm Form* (DKNF) jika setiap batasan dapat disimpulkan secara sederhana dengan mengetahui sekumpulan nama atribut dan domainnya selama menggunkan sekumpulan atribut pada kuncinya.

II.6.1. Tujuan Normalisasi

Tujuan utama normalisasi adalah mengidentifikasi kesesuaian hubungan yang mendukung data untuk memenuhi kebutuhan perusahaan. Adapun karakteristik hubungan tersebut mencakup (Indrajani, 2015, hal. 7).

1. Minimal jumlah atribut yang diperlukan untuk mendukung kebutuhan perusahaan
2. Atribut dengan hubungan logika yang menjelaskan mengenai *functional dependencies.*
3. Minimal duplikasi untuk tiap atribut.

II.7. *Unified Modeling Language* (UML)

UML adalah salah satu alat bantu yang sangat handal di dunia pengembangan sistem yang berorientasi objek, UML dapat menyediakan bahasa pemodelan yang mudah dimengerti oleh pengembang dan dapat dikomunikasikan dengan pemakai (Sopiah, 2011, hal. 3).

Adapun diagram-diagram yang dibuat pada UML yang terdiri dari:

1) Diagram perilaku (diagram *use-case* (*use case diagram*), diagram sekuen (*sequence diagram*), diagram kolaborasi (*collaboration diagram*), diagram statechart (*statechart diagram*) dan diagram aktivitas (*activity diagram*)).
2) Diagram struktur (diagram kelas (*class diagram*), diagram objek (*object diagram*), diagram komponen (*component diagram*) dan diagram *deployment* (*deployment diagram*))

II.7.1. Diagram UML (*Unified Modeling Language*)

Adapun diagram-diagram yang dibuat pada UML menurut Haryanto terdiri dari : (Sopiah, 2011, hal. 3).

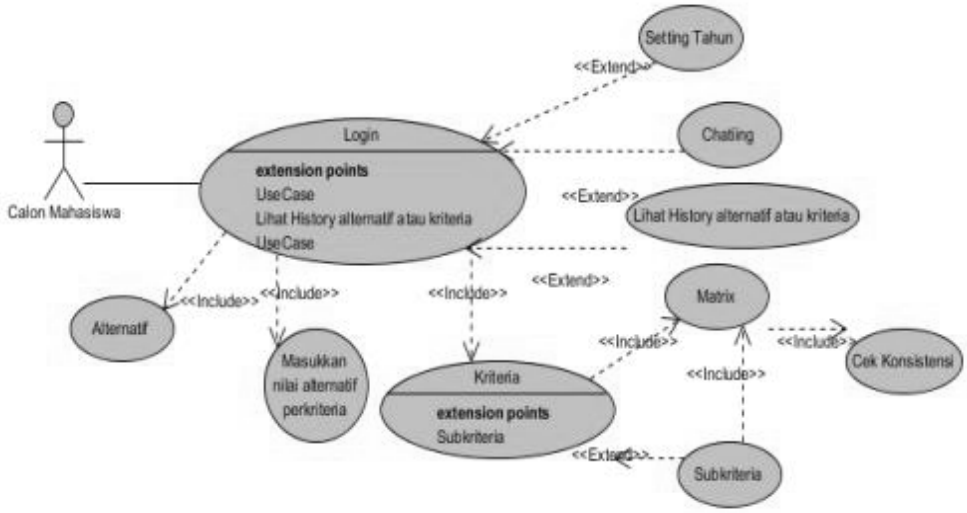
1) Diagram perilaku (diagram *use-case* (*usecase diagram*), diagram sekuen (*sequence diagram*), diagram kolaborasi (*collaboration diagram*), diagram statechart (*statechart diagram*) dan diagram aktivitas (*activity diagram*)).
2) Diagram struktur (diagram kelas (*class diagram*), diagram objek (*object diagram*), diagram komponen (*component diagram*) dan diagram *deployment* (*deployment diagram*)).

UML memiliki beberapa jenis diagram, namun dalam penulisan skripsi ini penulis hanya menggunakan 4 jenis diagram UML yaitu : *Use Case Diagram*,

Class Diagram, Activity Diagram dan Sequence Diagram. Adapun penjelasan dari diagram-diagram berikut adalah:

1. *Use case diagram*

   Use case model adalah teknik pemodelan untuk mendapatkan *functional requirement* dari sebuah sistem, menggambarkan interaksi antara pengguna dan sistem, menjelaskan secara naratif bagaimana sistem akan digunakan, menggunakan skenario untuk menjelaskan setiap aktivitas yang mungkin terjadi. Ada beberapa bagian didalam use case model.



Gambar II.2. Use Case Diagram
(Sumber : Norhikmah dkk, 2014 : 166)

a. *Use Case*, untuk mengetahui action atau prosedur apa yang ada didalam sistem.

b. *Actor*, siapa saja yang terlibat dalam action tersebut.

c. *Relationship*, bagaimana actions saling berelasi satu sama lain didalam sistem.

2. *Class diagram*

Class diagram merupakan diagram paling umum yang dijumpai dalam pemodelan berbasis UML. Didalam Class diagram terdapat class dan interface beserta atribut-atribut dan operasinya, relasi yang terjadi antar objek, *constraint* terhadap objek-objek yang saling berhubungan dan *inheritance* untuk organisasi class yang lebih baik. Class diagram juga terdapat *static view* dari elemen pembangun sistem. Pada intinya Class diagram mampu membantu proses pembuatan sistem dengan memanfaatkan konsep *forward* ataupun *reverse engineering.*

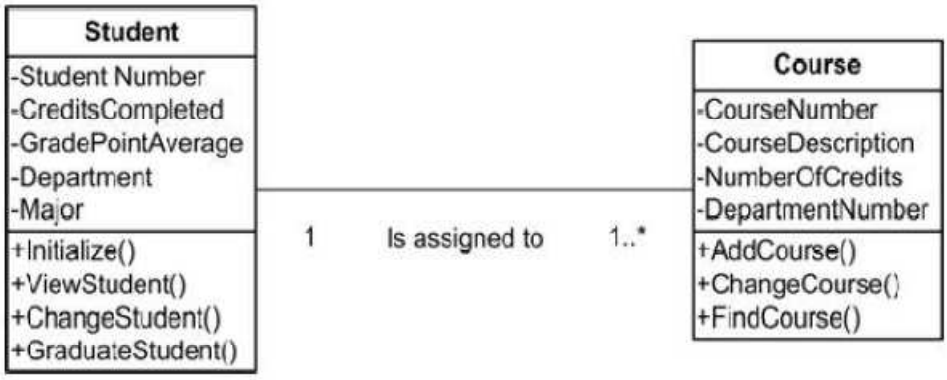
Class diagram mempunyai 2 komponen penting, yaitu:

1. *Structural*, yaitu ciri pembeda objek.
2. *Behavioral,* yaitu tingkah laku atau kegiatan yang mampu dilakukan

Berbagai simbol yang hadir didalam class diagram antara lain adalah:

1. *Class*, yang berfungsi untuk merepresentasikan tipe dari data yang dimilikinya. Class diagram dapat ditampilkan dengan menunjukkan atribut dan operasi yang dimilikinya atau hanya menunjukkan nama class-nya saja. Dapat juga kita tuliskan nama class dengan atributnya saja atau nama class dengan operasinya.
2. *Attribute*, merupakan data yang terdapat didalam class dan instance-nya dengan operator.
3. *Operation*, berfungsi untuk merepresentasikan fungsi-fungsi yang ditampilkan oleh class dan instance-nya dengan operator.

4. *Association*, digunakan untuk menunjukkan bagaimana dua class berhubungan satu sama lainnya. Association ditunjukkan dengan sebuah garis yang terletak diantara dua class. Didalam setiap association terdapat multiplicity, yaitu simbol yang mengindikasikan berapa banyak instance dari class pada ujung association yang satu dengan instance class di ujung association lainnya.
5. *Generalizations*, berfungsi untuk mengelompokkan class ke dalam hirarki inheritance.
6. *Aggregation*, merupakan bentuk khusus dari association yang merepresentasikan hubungan “part-whole”. Bagian “whole” dari hubungan ini sering disebut dengan assembly atau aggregate. Class yang satu dapat dikatakan merupakan bagian dari class yang lain yang ikut membentuk class tersebut.
7. *Composition*, merupakan jenis aggregation yang lebih kuat diantara dua class yang memiliki association dimana jika *whole* ditiadakan, maka *part*-nya juga ikut ditiadakan. Berbeda dengan aggregation, *part* akan tetap bisa berdiri sendiri meskipun bagian *whole*-nya ditiadakan.
8. Penggunaan operator (+) dalam class diagram diartikan dengan public, operator (-) diartikan private, dan operator (#) diartikan protected.



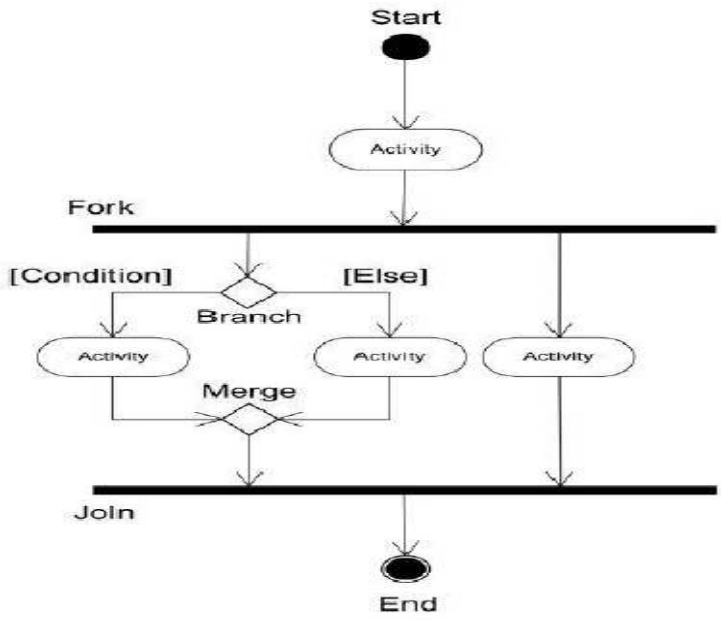
Gambar II.3. Notasi Class Diagram

(Sumber : Edgar Winata, Johan Setiawan ; 2013; 38)

3. *Activity diagram*

Teknik untuk menjelaskan *business process*, *procedural logic*, dan *work flow*. Bisa dipakai untuk menjelaskan teks use case dalam notasi grafis dengan menggunakan notasi yang mirip *flow chart*, meskipun terdapat sedikit perbedaan notasi :

1. *Nodes*, menandakan initial dan final node, final node boleh lebih dari 1.
2. *Activity*, aktivitas sistem dapat berupa aktivitas fisik juga bagi user.
3. *Flow/edge*, arah sebuah proses.
4. *Fork*, awal sebuah proses paralel.
5. *Join* akhir proses paralel.
6. *Condition*, kondisi yang dituliskan dalam bentuk teks
7. *Decision*, implementasi if dan then.
8. *Merge*, penyatuan beberapa flow.
9. *Partition*, siapa atau apa yang menjalankan aktivitas.



Gambar II.4. Notasi Activity Diagram
(Sumber : Edgar Winata, Johan Setiawan ; 2013)

4. *Sequence diagram*

   Menjelaskan interaksi obyek-obyek yang saling berkolaborasi (berhubungan), mirip dengan *activity diagram* yaitu menggambarkan alur kejadian sebuah aktivitas tetapi lebih detil dalam menggambarkan aliran data termasuk data atau behaviour yang dikirimkan atau diterima namun kurang mampu menjelaskan detil dari sebuah algoritma.

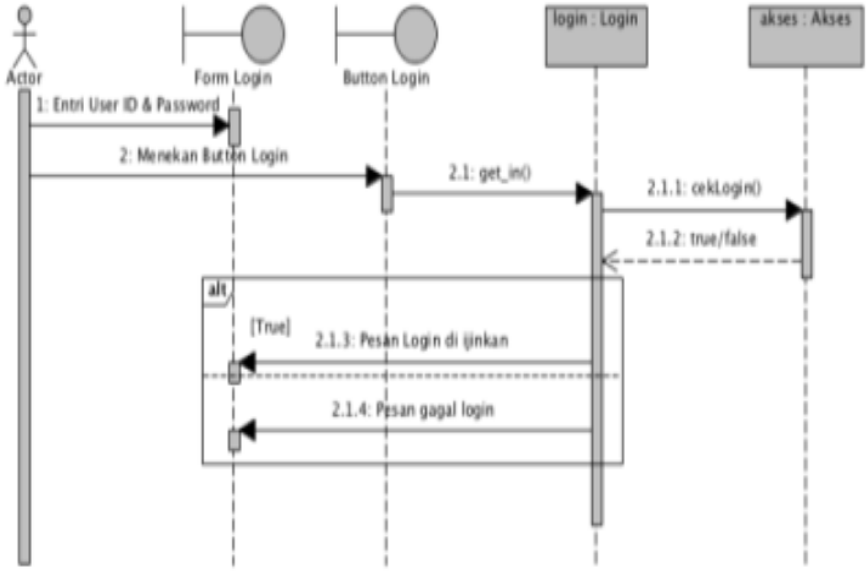
   Dalam *sequence diagram* terdapat beberapa bagian.

   1. *Participant*, yaitu objek yang terkait dengan sebuah urutan proses.

   2. *Lifeline*, menggambarkan daur hidup sebuah objek.

   3.*Activation*, suatu titik waktu dimana sebuah objek mulai berpartisipasi dalam sebuah sequence.

4. *Time*, elemen paling penting dalam sequence diagram yang konteksnya adalah urutan, bukan durasi.
5. *Return*, suatu hasil kembalian sebuah operasi. Operasi mengembalikan hasil tetapi boleh tidak ditulis jika tidak ada perbedaan dengan Getter-nya.



Gambar II.5. Notasi Sequence Diagram
(Sumber : Norhikmah dkk, 2014; 166)

Untuk menggambarkan analisa dan desain diagram, UML memiliki seperangkat notasi yang akan digunakan ke dalam tiga kategori diatas yaitu struktur diagram, behaviour diagram, dan interaction diagram. Berikut beberapa notasi dalam UML diantaranya :

1) *Actor*, menentukan peran yang dimainkan oleh user atau sistem lain yang berinteraksi dengan subjek. *Actor* adalah segala sesuatu yang berinteraksi langsung dengan sistem aplikasi komputer, seperti orang, benda atau lainnya. Tugas actor adalah memberikan

informasi kepada sistem dan dapat memerintahkan sistem untuk melakukan sesuatu tugas.

2) *Class diagram,* Notasi utama dan yang paling mendasar pada diagram UML adalah notasi untuk mempresentasikan suatu class beserta dengan atribut dan operasinya. Class adalah pembentuk utama dari sistem berorientasi objek.

3) *Use Case* dan *use case specification, Use case* adalah deskripsi fungsi dari sebuah sistem perspektif pengguna. Use case bekerja dengan cara mendeskripsikan tipikal interaksi antara user (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Urutan langkah-langkah yang menerangkan antara pengguna dan sistem disebut skenario. Use case merupakan awal yang sangat baik untuk setiap fase pengembangan berbasis objek, design, testing, dan dokumentasi yang menggambarkan kebutuhan sistem dari sudut pandang di luar sistem. Perlu diingat bahwa use case hanya menetapkan apa yang seharusnya dikerjakan oleh sistem, yaitu kebutuhan fungsional sistem dan tidak untuk menentukan kebutuhan nonfungsional, misalnya: sasaran kinerja, bahasa pemrograman dan lain sebagainya.

4) *Interaction, Interaction* digunakan untuk menunjukkan baik aliran pesan atau informasi antar obyek maupun hubungan antar obyek.

5) *Association, Association* menggambarkan navigasi antar *class (navigation*), berapa banyak obyek lain yang bisa berhubungan dengan satu obyek *(multiplicity antar class)* dan apakah suatu class menjadi bagian dari class lainnya *(aggregation).*

II.8. *Microsoft Visual Basic 2010*

Visual Basic .NET adalah Visual Basic yang direkayasa kembali untuk digunakan pada *platform* .NET sehingga aplikasi yang dibuat menggunakan Visual Basic .NET dapat berjalan pada sistem komputer apa pun, dan dapat mengambil data dari *server* dengan tipe apa pun asalkan terinstal .NET Framework. Pada umumnya Visual Basic .NET terpaket dalam Visual Studio .NET. visual studio .NET yaitu versi Professional, Premium dan yang paling lengkap adalah versi Ultimate. Pada intinya, Visual Basic .NET adalah salah satu dari kumpulan *tools* pemograman yang terdapat pada paket Visual Studio .NET. pada Visual Studio .NET terdapat beberapa *tools* pemograman lain seperti C++ .NET, Visual C# .NET, dan Visual J# .NET (Hidayatullah, 2012, hal. 5).

II.9. *Microsoft SQL Server 2008*

Structured Query Language merupakan bahasa yang digunakan untuk mengakses data dalam basis data yang relasional. Bahasa ini merupakan bahasa standar yang digunakan dalam manajemen basis data relasional. Secara umum, SQL terdiri dari dua bahasa, yaitu *Data Definition Language* (DDL) dan *Data Manipulation Language* (DML). DDL digunakan untuk mendefinisikan, mengubah, serta menghapus *database* dan objek yang diperlukan dalam *database*

seperti tabel, view, user, dan sebagainya. DDL yang umum digunakan adalah CREATE untuk membuat objek baru, ALTER untuk mengubah objek yang sudah ada, dan DROP untuk menghapus objek. SQL Server adalah sebuah DBMS (Database Management System) yang dibuat oleh Microsoft untuk ikut berkecimpung dalam persaingan dunia pengolahan data menyusul pendahulunya seperti IBM dan Oracle.

Microsoft SQL Server adalah sebuah sistem manajemen basis data relasional (RDBMS) produk microsoft. Bahasa query utamanya adalah Transact-SQL yang merupakan implementasi dari SQL standar ANSI/ISO yang digunakan oleh Microsoft dan Sybase (Triyanto, Widada, & Hariyati, 2011, hal. 43).