BAB II
LANDASAN TEORI

II.1. Sistem Pendukung Keputusan (SPK)

Pada dasarnya sistem pendukung keputusan merupakan pengembangan lebih lanjut dari sistem informasi manajemen terkomputerisasi yang dirancang sedemikian rupa sehingga bersifat interaktif dengan pemakainya. Sifat interaktif dimaksudkan untuk memudahkan integrasi antara berbagai komponen dalam proses pengambilan keputusan seperti prosedur, kebijakan, teknik analisis, serta pengalaman dan wawasan manajerial guna membentuk suatu kerangka keputusan bersifat fleksibel. Konsep Sistem Pendukung Keputusan (SPK)/*Decision Support Sistem* (DSS) pertama kali diungkapkan pada awal tahun 1970-an oleh Michael S. Scott Morton dengan istilah *Management Decision Sistem*. Sistem tersebut adalah suatu sistem yang berbasis komputer yang ditujukan untuk membantu pengambil keputusan dengan memanfaatkan data dan model tertentu untuk memecahkan berbagai persoalan yang tidak terstruktur.(Desi Leha Kurniasih : jurnal Pelita Informatika Budi Darma, vol III No : 2, April 2013).

II.1.1 Ciri-ciri Sistem Pendukung Keputusan (SPK)

Menurut Kosasi dan Kusrini (2007), adapun ciriciri sebuah SPK seperti yang dirumuskan oleh Alters Keen adalah sebagai berikut:

1. SPK ditujukan untuk membantu pengambilan keputusan-keputusan yang kurang terstruktur dan umumnya dihadapi oleh para manajer yang berada di tingkat puncak.

2. SPK merupakan gabungan antara kumpulan model kualitatif dan kumpulan data.
3. SPK memiliki fasilitas interaktif yang dapat mempermudah hubungan antara manusia dengan komputer.
4. SPK bersifat luwes dan dapat menyesuaikan dengan perubahan-perubahan yang terjadi.

II.1.2 Karakteristik, Kemampuan, dan Keterbatasan SPK

Sehubungan banyaknya definisi yang dikemukakan mengenai pengertian dan penerapan dari sebuah SPK, sehingga menyebabkan terdapat banyak sekali pandangan mengenai sistem tersebut. Selanjutnya Turban (1996), menjelaskan terdapat sejumlah karakteristik dan kemampuan dari SPK yaitu:

A. Karakteristik SPK

1. Mendukung seluruh kegiatan organisasi
2. Mendukung beberapa keputusan yang saling berinteraksi
3. Dapat digunakan berulang kali dan bersifat konstan
4. Terdapat dua komponen utama, yaitu data dan model
5. Menggunakan baik data eksternal dan internal
6. Memiliki kemampuan *what-if analysis* dan *goal seeking analysis*
7. Menggunakan beberapa model kuantitatif

B. Kemampuan SPK

1. Menunjang pembuatan keputusan manajemen dalam menangani masalah semi terstruktur dan tidak terstruktur

2. Membantu manajer pada berbagai tingkatan manajemen, mulai dari manajemen tingkat atas sampai manajemen tingkat bawah
3. Menunjang pembuatan keputusan secara kelompok maupun perorangan
4. Menunjang pembuatan keputusan yang saling bergantung dan berurutan
5. Menunjang tahap-tahap pembuatan keputusan antara lain *intelligensi, desain, choice, dan implementation*
6. Kemampuan untuk melakukan adaptasi setiap saat dan bersifat fleksibel
7. Kemudahan melakukan interaksi sistem
8. Meningkatkan efektivitas dalam pembuatan keputusan dari pada efisiensi
9. Mudah dikembangkan oleh pemakai akhir
10. Kemampuan pemodelan dan analisis pembuatan keputusan
11. Kemudahan melakukan pengaksesan berbagai sumber dan format data

C. Keterbatasan SPK

1. Ada beberapa kemampuan manajemen dan bakat manusia yang tidak dapat dimodelkan, sehingga model yang ada dalam sistem tidak semuanya mencerminkan persoalan sebenarnya.
2. Kemampuan suatu SPK terbatas pada pembendaharaan pengetahuan yang dimilikinya (pengetahuan dasar serta model dasar).
3. Proses-proses yang dapat dilakukan oleh SPK biasanya tergantung juga pada kemampuan perangkat lunak yang digunakannya.

SPK tidak memiliki kemampuan intuisi seperti yang dimiliki oleh manusia. Karena walau bagaimana pun canggihnya suatu SPK, hanyalah sautu

kumpulan perangkat keras, perangakat lunak dan sistem operasi yang tidak dilengkapi dengan kemampuan berpikir.

II.1.3 Tahapan Sistem Pengambilan Keputusan

Menurut Herbert A. Simon ada 4 tahap yang harus dilalui dalam proses pengambilan keputusan

yaitu :

1. Penelusuran (*intelligence*)

Tahap ini merupakan tahap pendefinisian masalah serta identifikasi informasi yang dibutuhkan yang berkaitan dengan persoalan yang dihadapi serta keputusan yang akan diambil.

2. Perancangan (*design*)

Tahap ini merupakan tahap analisa dalam kaitan mencari atau merumuskan alternatif-alternatif pemecahan masalah.

3. Pemilihan (*choice*)

Yaitu memilih alternatif solusi yang diperkirakan paling sesuai.

4. Implementasi (*implementation*)

Tahap ini merupakan tahap pelaksanaan dari keputusan yang telah diambil.

II.2. Pengertian Metode TOPSIS

TOPSIS adalah salah satu metode pengambilan keputusan multikriteria yang pertama kali diperkenalkan oleh Yoon dan Hwang (1981). TOPSIS menggunakan prinsip bahwa alternatif yang terpilih harus mempunyai jarak

terdekat dari solusi ideal positif dan terjauh dari solusi ideal negatif dari sudut pandang geometris dengan menggunakan jarak Euclidean untuk menentukan kedekatan relatif dari suatu alternatif dengan solusi optimal. Solusi ideal positif didefinisikan sebagai jumlah dari seluruh nilai terbaik yang dapat dicapai untuk setiap atribut, sedangkan solusi negatif-ideal terdiri dari seluruh nilai terburuk yang dicapai untuk setiap atribut. TOPSIS mempertimbangkan keduanya, jarak terhadap solusi ideal positif dan jarak terhadap solusi ideal negatif dengan mengambil kedekatan relatif terhadap solusi ideal positif. Berdasarkan perbandingan terhadap jarak relatifnya, susunan prioritas alternatif bisa dicapai. Metode ini banyak digunakan untuk menyelesaikan pengambilan keputusan. Hal ini disebabkan konsepnya sederhana, mudah dipahami, komputasinya efisien, dan memiliki kemampuan mengukur kinerja relatif dari alternatif-alternatif keputusan.

II.2.1. Langkah-Langkah Metode TOPSIS

1. Membangun *normalized decision matrix*

Elemen *rij* hasil dari normalisasi *decision matrix R* dengan metode *Euclidean length of a vector* adalah :

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x_{ij}^2}}$$

Gambar II.1. Rumus Normalisasi Matriks Keputusan R
(Sumber : Desi Leha Kurniasih ; Pelita Informatika Budi Darma ; 2013 : 8)

Dimana :

r ij = hasil dari normalisasi matriks keputusan R

i = 1,2,3,...,m;

j = 1,2,3,...,n;

2. Membangun *weighted normalized decision matrix*

Dengan bobot *W* = (*w1*,*w2*,....,*wn*), maka normalisasi bobot matriks *V* adalah :

$$V=\begin{bmatrix} w_1r_{11} & w_2r_{12} & \dots & w_nr_{1n} \\ w_1r_{21} & & & \\ \cdot & & & \\ w_1m_1 & w_2r_{m2} & \dots & w_nr_{mn} \end{bmatrix}$$

Gambar II.2. Rumus Normalisasi Bobot Matriks V
(Sumber : Desi Leha Kurniasih ; Pelita Informatika Budi Darma ; 2013 : 8)

3. Menentukan solusi ideal positif dan solusi ideal negatif

Solusi ideal positif dinotasikan dengan A+ dan solusi ideal negatif dinotasikan dengan A-, sebagai berikut :

$$A^* = \{ (\max v_{ij} | j \in J), (\min v_{ij} | j \in J'), i = 1, 2, 3, \dots, m \} = \{v_{1*}, v_{2*}, \dots, v_{n*}\}$$
$$A^- = \{ (\min v_{ij} | j \in J), (\max v_{ij} | j \in J'), i = 1, 2, 3, \dots, m \} = \{v_1^-, v_2^-, \dots, v_n^-\}$$

Gambar II.3. Rumus Menentukan Solusi Ideal (+) dan (-)
(Sumber : Desi Leha Kurniasih ; Pelita Informatika Budi Darma ; 2013 : 8)

Dimana :

V_{ij} = elemen matriks *V* baris ke-*i* dan kolom ke- *j*

J = {j=1,2,3,...,n dan j berhubungan dengan *benefit criteria*}

J' = {j=1,2,3,...,n dan j berhubungan dengan *cost criteria*}

4. Menghitung separasi

Separation measure ini merupakan pengukuran jarak dari suatu alternatif ke solusi ideal positif dan solusi ideal negatif. Perhitungan matematisnya adalah sebagai berikut :

$$S_{i*} = \sqrt{\sum_{j=1}^{n} \left(v_{ij} - v_j^*\right)^2}, \text{dengan } i=1,2,3,\ldots,m$$

Gambar II.4. Rumus *Separation Measure* Untuk Solusi Ideal Positif
(Sumber : Desi Leha Kurniasih ; Pelita Informatika Budi Darma ; 2013 : 8)

Dimana :

J = {j=1,2,3,...,n dan j merupakan *benefit criteria*}

J' = {j=1,2,3,...,n dan j merupakan *cost criteria*}

$$S_{i-} = \sqrt{\sum_{j=1}^{n} \left(v_{ij} - v_j^-\right)^2}, \text{dengan } i=1,2,3,\ldots,m$$

Gambar II.5. Rumus *Separation Measure* Untuk Solusi Ideal Negatif
(Sumber : Desi Leha Kurniasih ; Pelita Informatika Budi Darma ; 2013 : 8)

Dimana :

J = {j=1,2,3,...,n dan j merupakan *benefit criteria*}

J' = {j=1,2,3,...,n dan j merupakan *cost criteria*}

5. Menghitung kedekatan relatif terhadap solusi ideal

Kedekatan relatif dari alternatif A+ dengan solusi ideal A- direpresentasikan dengan :

$$C_{i*} = \frac{S_{i-}}{S_{i*} + S_{i-}}, \text{dengan } 0 < C_{i*} < 1 \text{ dan } i=1,2,3,\ldots,m$$

Gambar II.6 Rumus Menghitung Kedekatan Relatif Terhadap Solusi Ideal
(Sumber : Desi Leha Kurniasih ; Pelita Informatika Budi Darma ; 2013 : 9)

6. Merangking alternatif

Alternatif dapat diranking berdasarkan urutan *Ci**. Maka dari itu, alternatif terbaik adalah salah satu yang berjarak terpendek terhadap solusi ideal dan berjarak terjauh dengan solusi ideal.

II.3. *Microsoft Visual Studio* 2010

Visual Basic diturunkan dari bahasa *BASIC*. *Visual Basic* terkenal sebagai bahasa pemograman yang mudah untuk digunakan terutama untuk memebuat aplikasi yang berjalan di atas *platform* Windows. Pada tahun 90an, *Visual Basic* menjadi bahasa pemograman yang paling populer dan menjadi pilihan utama untuk mengembangkan program berbasis windows . Versi *Visual Basic* terakhir sebelum berjalan diatas *.NET Framework* adalah VB6 (*Visual Studio* 1998). (ANDI Yogyakarta ; 2010 : 1)

Visual Basic .NET dirilis pada bulan februari tahun 2002 bersamaan dengan *platform .NET Framework* 1.0. Kini sudah ada beberapa versi dari *Visual Basic* yang berjalan pada *platform .NET* , yaitu VB 2002 (VB7), VB 2005 (VB8), VB 2008 (VB9), dan yang terakhir adalah VB 2010 (VB10) yang dirilis bersamaan dengan *Visual Studio* 2010. Selain *Visual Basic* 2010, Visual Studio 2010 juga mendukung beberapa bahasa lain, yaitu C#, C++, F# (bahasa baru untuk *functional programming*), *IronPhyton*, dan *IronRuby* (bahasa baru untuk *dynamic programming*). (ANDI Yogyakarta ; 2010 : 1)

II.4. *Microsoft SQL Server* 2008

SQL Server 2008 adalah sebuah terobosan baru dari *Microsoft* dalam bidang *database*. *SQL Server* adalah sebuah DBMS (*Database Management System*) yang dibuat oleh *Microsoft* untuk ikut berkecimpung dalam persaingan dunia pengolahan data menyusul pendahuluannya seperti *IBM* dan *Oracle*. *SQL Server* 2008 dibuat pada saat kemajuan dalam bidang *hardware* sedemikian pesat. Oleh karena itu sudah dapat dipastikan bahwa *SQL Server* 2008 membawa beberapa terobosan dalam bidang pengolahan dan penyimpanan data. (Wahana Komputer ; 2010 : 2).

Terdapat 3 jenis perintah *SQL* yaitu (Achmad Solichin ; 2010 : 35) :

a. DDL atau *Data Definition Language*

   DDL merupakan perintah *SQL* yang berhubungan dengan pendefinisian suatu struktur *database*, dalam hal ini *database* dan *table*. Beberapa perintah dasar yang termasuk DDL ini antara lain :

   - *CREATE*
   - *ALTER*
   - *RENAME*
   - *DROP*

b. DML atau *Data Manipulation Language*

   DDL merupakan perintah *SQL* yang berhubungan dengan manipulasi atau pengolahan data atau *record* dalam *table*. Perintah *SQL* yang termasuk dalam DML antara lain :

   - *SELECT*

- *INSERT*
- *UPDATE*
- *DELETE*

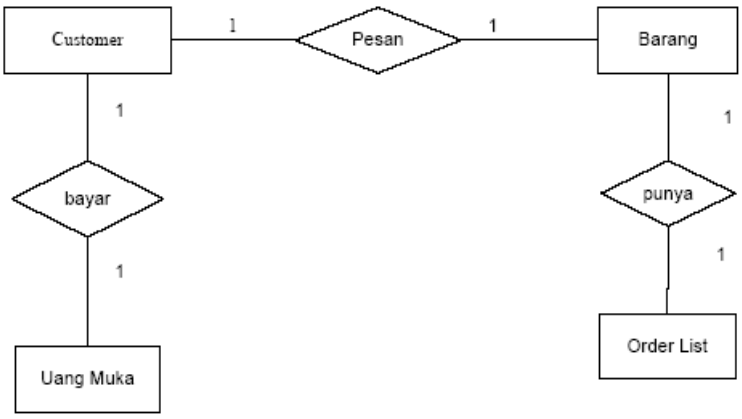
c. DCL atau *Data Control Language*

DCL merupakan perintah *SQL* yang berhubungan dengan manipulasi user dan hak akses (*priviledges*). Perintah *SQL* yang termasuk dalam DCL antara lain :

- *GRANT*
- *REVOKE*

II.5. *Entity Relationship Diagram* (ERD)

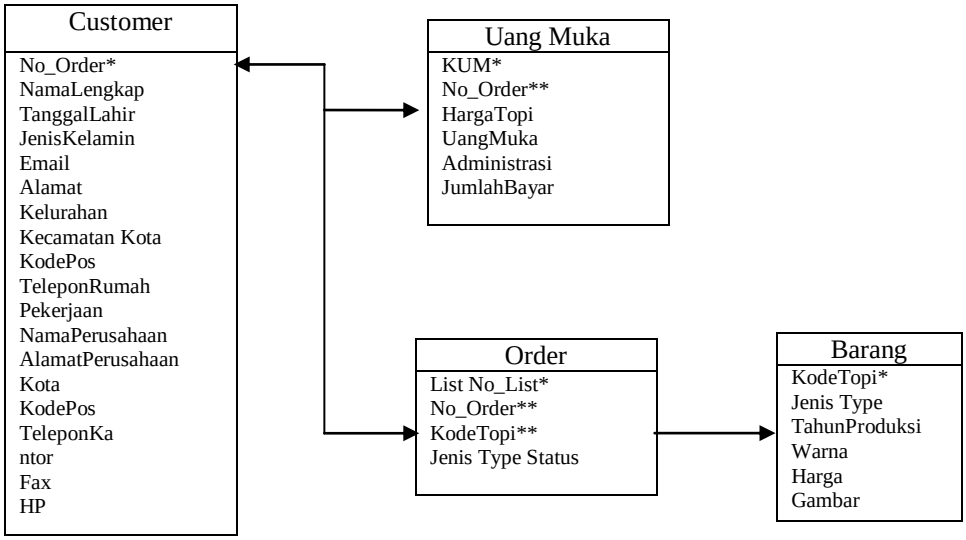
Entity Relationship Diagram atau diagram hubungan entitas dari sistem penjualan yang diusulkan berfungsi untuk menggambarkan model basis data yang akan dipakai. Model basis data yang digunakan adalah basis data relasional, dimana setiap entitas saling memiliki hubungan dengan entitas lain. (Iyan Gustiana ; 2010 : 8).



Gambar II.7. Contoh *Entity Relationship Diagram*
(Sumber : Iyan Gustiana ; 2010 : 9)

II.6. Normalisasi

Normalisasi dilakukan agar basis data yang akan diterapkan dapat digunakan dan dioperasikan dengan efesien, mudah dan tidak mengalami anomali atau keanehan. Normalisasi dimulai dengan menganalisa tabel dalam bentuk tidak normal. (Iyan Gustiana ; 2010 : 9).



Gambar II.8. Contoh Normalisasi
(Sumber : Iyan Gustiana ; 2010 : 9)

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

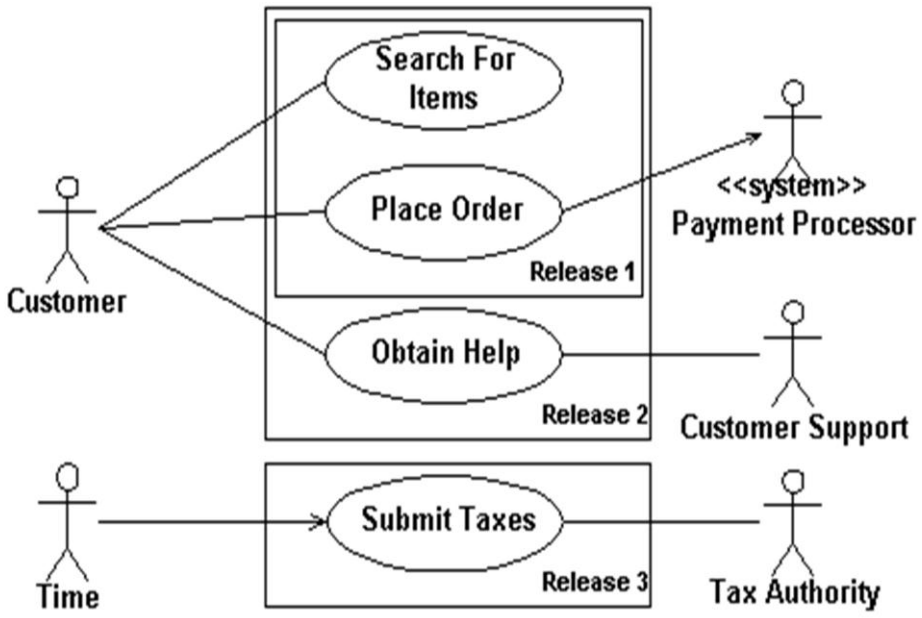
Unified Modelling Language (UML) adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk *visualisasi*, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. (Yuni Sugiarti ; 2013 ; 34).

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 pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa berorientasi objek seperti *C++*, *Java*, *C#* atau *VB.NET*. Walaupun demikian, UML tetap dapat digunakan untuk modeling apllikasi procedural dalam *VB* atau C. (Yuni Sugiarti ; 2013 ; 34).

Unified Modelling Language (UML) adalah suatu alat untuk memvisualisasikan dan mendokumentasikan hasil analisa dan desain yang berisi sintak dalam memodelkan sistem secara *visual*. Juga merupakan satu kumpulan konvensi pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem *software* yang terkait dengan objek. (Jurnal Informatika Mulawarman ; Haviluddin ; 2011 : 1).

II.7.1. *Use Case Diagram*

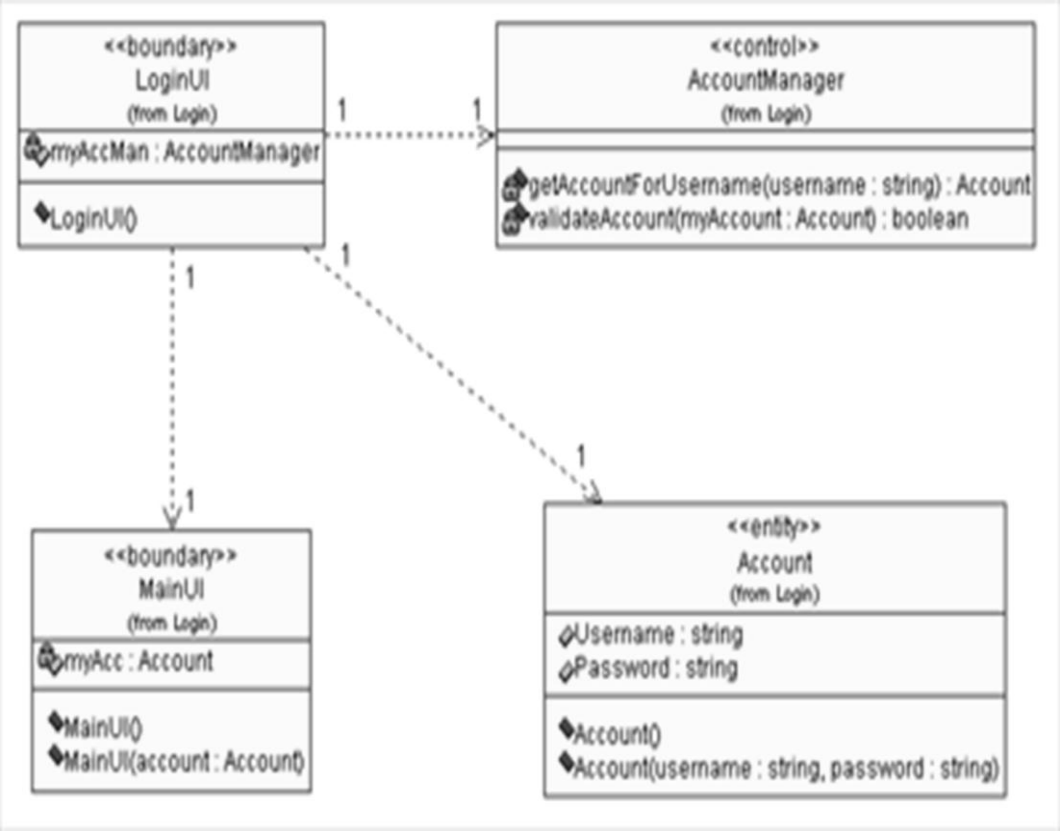
Diagram yang menggambarkan *actor*, *use case* dan relasinya sebagai suatu urutan tindakan yang memberikan nilai terukur untuk aktor. Sebuah *use case* digambarkan sebagai *elips horizontal* dalam suatu diagram UML *use case*. (Haviluddin ; Jurnal Informatika Mulawarman ; 2011 : 6)



Gambar II.9. Contoh *Use Case Diagram*
(Sumber : Haviluddin ; Jurnal Informatika Mulawarman ; 2011 : 6)

II.7.2. *Class Diagram*

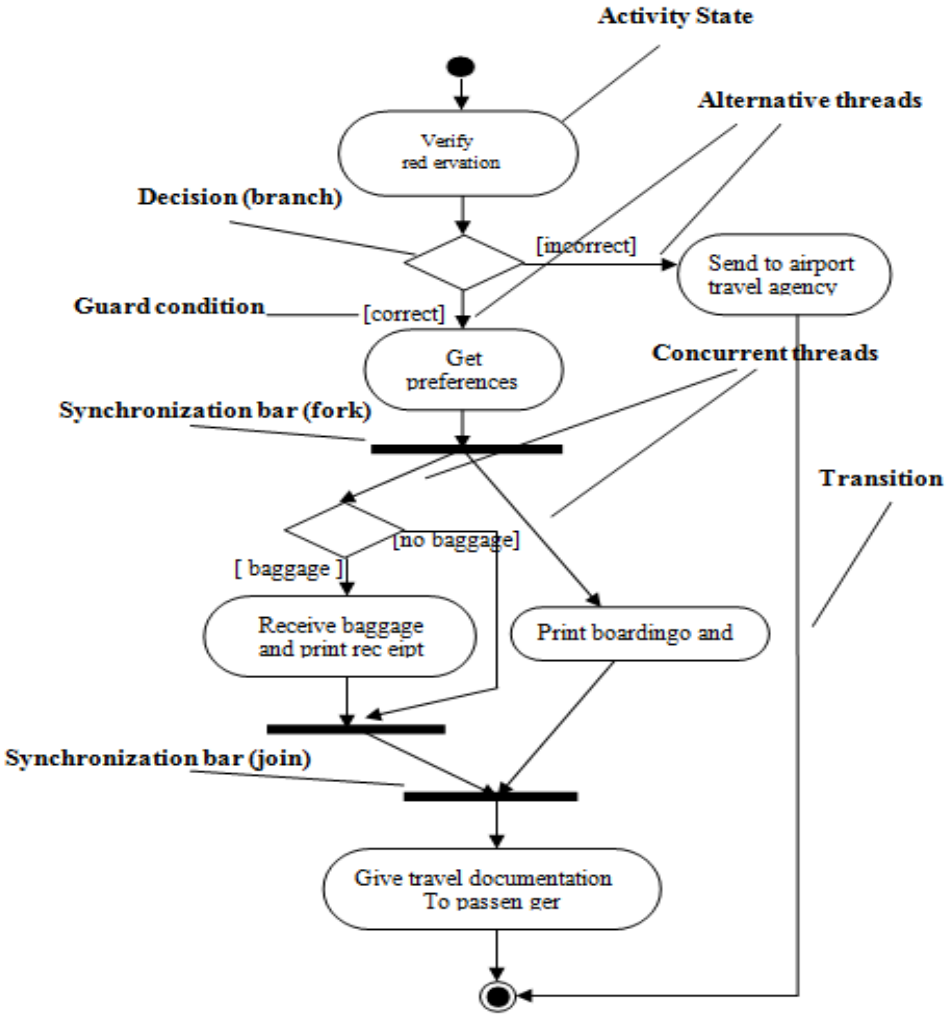
Class diagram menggambarkan struktur statis dari kelas dalam sistem anda dan menggambarkan atribut, operasi dan hubungan antara kelas. *Class diagram* membantu dalam memvisualisasikan struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak dipakai.Selama tahap desain, *class diagram* berperan dalam menangkap struktur dari semua kelas yang membentuk arsitektur sistem yang dibuat.(Haviludin ; Jurnal Informatika Mulawarman ; 2011 ; 3)



Gambar II.10. Contoh *Class Diagram*
(Sumber : Haviludin ; Jurnal Informatika Mulawarman ; 2011 ; 3)

II.7.3. *Activity Diagram*

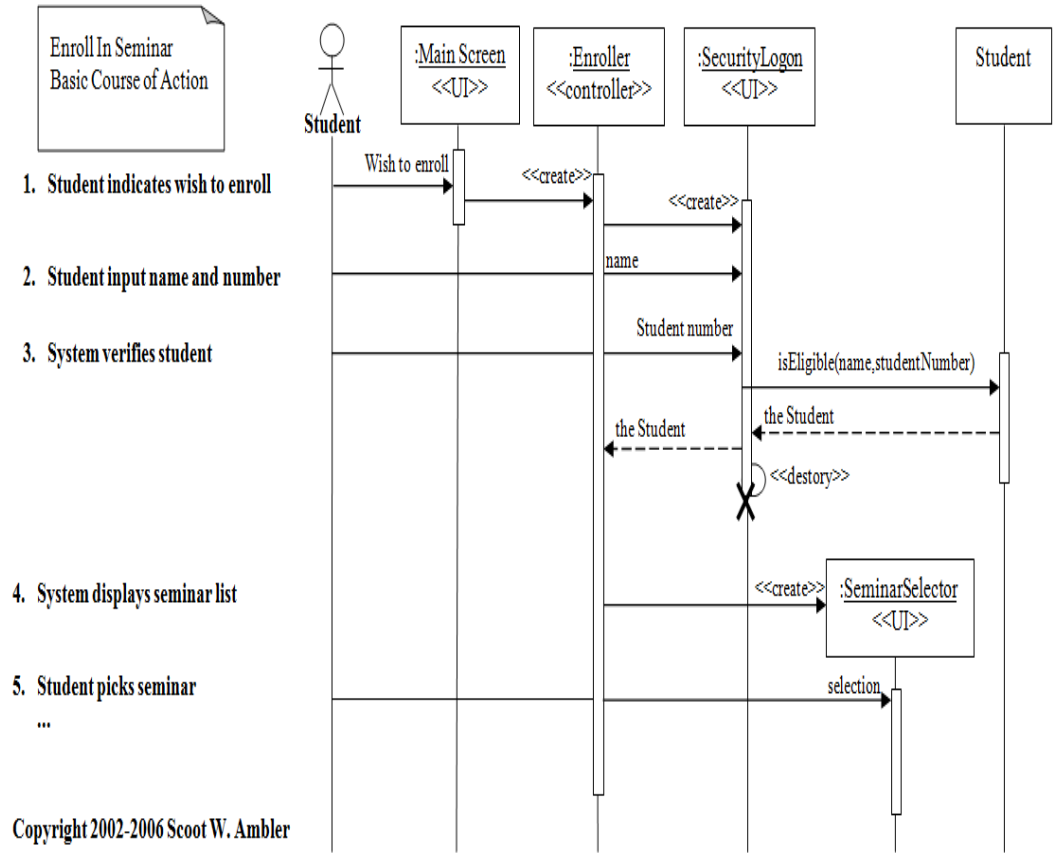
Menggambarkan aktifitas-aktifitas, objek, *state*, transisi *state* dan *event*. Dengan kata lain kegiatan diagram alur kerja menggambarkan perilaku sistem untuk aktifitas.(Haviludin ; Jurnal Informatika Mulawarman ; 2011 ; 4)



Gambar II.11. Contoh *Activity Diagram*
(Sumber : Haviludin ; Jurnal Informatika Mulawarman ; 2011 ; 4)

II.7.4. *Sequence Diagram*

Sequence diagram menjelaskan interaksi objek yang disusun berdasarkan urutan waktu. Secara mudahnya *sequence diagram* adalah gambaran tahap demi tahap, termasuk kronologi (urutan) perubahan secara logis yang seharusnya dilakukan untuk menghasilkan sesuatu sesuai dengan *use case* diagram. (Haviludin ; Jurnal Informatika Mulawarman ; 2011 ; 5)



Gambar II.12. Contoh *Sequence Diagram*
(Sumber : Haviludin ; Jurnal Informatika Mulawarman ; 2011 ; 5)