BAB II

LANDASAN TEORI

II.1. Sistem Pendukung Keputusan (DSS)

II.1.1. Pengertian Sistem Pendukung Keputusan

Decision Support System (DSS) adalah sistem informasi yang mendukung bisnis atau kegiatan organisasi yang melibatkan pengambilan keputusan. DSS sangat berguna dalam situasi yang cepat berubah dan sulit menentukan suatu kondisi yang belum pernah ditemui sebelumnya (arXiv ; Krzywicki : 2014 ; 4).

Sistem pendukung keputusan atau *Decision SupportSystems* (DSS) adalah sistem informasi interkatif yang menyediakan informasi, pemodelan, dan pemanipulasian data yang digunakan untuk membantu pengambilan keputusan pada siuasi yang semiterstruktur dan situasi yang tidak terstruktur di mana tak seorang pun tahu secara pasti bagaimana keputusan seharusnya dibuat. Konsep DSS dikemukaakan pertama kali oleh ScootMortonpada tahun 1971 (Turban, McLean, dan Wetherbe, 1999). Beliau mendefinisikan cikal bakal DSS tersebut sebagai : “sistem berbasis komputer yang interaktif, yang membantu pengambil keputusan dengan menggunakan data dan model untuk memecahkan persoalan-persoalan tak terstruktur (Pelita Informatika Budi Darma; Ahmad Khaidir : 2014 ; 150).

Sistem pendukung keputusan merupakan sistem berbasis komputer yang diharapkan dapat membantu menyelesaikan masalah-masalah yang komplek yang tidak terstruktur maupun yang semi terstruktur.Sistem pendukung keputusan

merupakan perpaduan antara keahlian manusia dan juga komputer. Dengan kemampuan yang dimiliki, sistem pendukung keputusan diharapkan dapat membantu dalam pengambilan keputusan baik untuk masalah semi terstruktur maupun tidak terstruktur (Jurnal TAM (Technology Acceptance Model) ;Erliza Septia Nagara : 2015 ; 4).

II.1.2. Karakteristik Sistem Pendukung Keputusan

Karakteristik dalam sistem pendukung keputusan, antara lain:

1. Sistem Pendukung Keputusan dirancang untuk membantu pengambil keputusan dalam memecahkan masalah yang sifatnya semi terstruktur ataupun tidak terstruktur dengan menambahkan kebijaksanaan manusia dan informasi komputerisasi.
2. Dalam proses pengolahannya, sistem pendukung keputusan mengkombinasikan penggunaan model-model analisis dengan teknik pemasukan data konvensional serta fungsi-fungsi pencari/interogasi informasi.
3. Sistem Pendukung Keputusan, dirancang sedemikian rupa sehingga dapat digunakan/dioperasikan dengan mudah.

Sistem Pendukung Keputusan dirancang dengan menekankan pada aspek fleksibilitas serta kemampuan adaptasi yang tinggi (Jurnal TAM (Technology Acceptance Model) ;Erliza Septia Nagara : 2015 ; 4).

II.1.3. Komponen Sistem Pendukung Keputusan

Untuk dapat menerapkan SPK, ada 4 komponen subsistem yang harus disediakan yaitu:

1. Subsistem manajemen data

   Subsistem ini menyediakan data bagi sistem, termasuk didalamnya basis data.Berisi data yang relevan untuk situasi dan diatur oleh perangkat lunak yang disebut *Database Management System* (DBMS).

2. Subsistem manajemen model

   Subsistem ini berfungsi sebagai pengelola berbagai model, mulai dari model keuangan, statistik, matematik, atau model kuantitatif lainnya yang memiliki kemampuan analisis dan manajemen perangkat lunakyangsesuai.Perangkat lunak ini sering disebut *Model Base Management System* (MBMS).

3. Subsistem manajemen pengetahuan

   Subsistem ini mendukung berbagai subsistem lainnya, atau dapat dikatakan berperan sebagai komponen yang independen.Subsistem ini menyediakan intelegensi untuk menambah pertimbangan pengambil keputusan.

4. Subsistem manajemen antar muka pengguna

   Subsistem ini berupa tampilan yang disediakan yang mampu mengintegrasikan sistem terpasang dengan pengguna secara interaktif. Melalui subsistem ini pengguna dapat berkomunikasi dengan sistem

pendukung keputusan serta memerintah sistem pendukung keputusan (JURNAL SAINS DAN SENI ITS ;Syukron Hidayat et al., : 2015 ; 44)

II.2. Metode Perbandingan Eksponensial (MPE)

Metode Perbandingan Eksponensial (MPE) merupakansalah satu metode untuk menentukan urutan prioritas alternatif keputusan dengan kriteriajamak. Dalam menggunakan Metode Perbandingan Eksponensial ada beberapa tahap yang harus dilakukan,yaitu: menyusun alternatif-alternatif keputusanyang akan dipilih, menentukan kriteria atau perbandingankeputusan yang penting untuk dievaluasi, menentukan tingkat kepentingan dari setiap kriteria keputusan,melakukan penilaian terhadap semua alternatif pada setiap kriteria, menghitung skor atau nilai total setiap alternatif, dan menentukan urutan prioritas keputusan didasarkan pada skor atau nilai total masing-masing alternatif (Diva Mahardika, 2012 ; 5-6).

Menurut Marimin, yang dikutip oleh Didie Nanda PribadiMetode Perbandingan*Eksponensial* (MPE) merupakan salah satu metode untuk menentukan urutan prioritas alternatif keputusan dengan kriteria jamak (PelitaInformatika Budi Darma ; Andri Januardi : 2013 ; 19).

Adapun rumus matematika yang dipakai dalam menggunakan Metode Perbandingan Ekponensial adalah:

Total Nilai TN I = $\sum_{j=1}^{m}(RKij)^{\mathrm{Bj}}$

Keterangan:

a. TNi : Total nilai alternatif ke-i
b. RKij : Derajat kepentingan relatif kriteria ke-j pada pilihan keputusan i
c. TKKj : Derajat kepentingan kriteria keputusan ke-j; TKKj > 0;
d. m : Jumlah kriteria keputusan
e. j : 1,2,3,…m; m : Jumlah kriteria
f. i :1,2,3,…,n; n : Jumlah pilihan alternatif

Dimana dalam menggunakan *Metode PerbandinganEksponensial* ada beberapa tahap yang harus dilakukan, yaitu:

1. Menyusun alternatif-alternatif keputusan yang akan dipilih.
2. Menentukan kriteria atau perbandingan keputusan yang penting untuk dievaluasi.
3. Menentukan tingkat kepentingan dari setiap kriteria keputusan.
4. Melakukan penilaian terhadap semua alternatif pada setiap kriteria.
5. Menghitung skor atau nilai total setiap alternative.
6. Menentukan urutan prioritas keputusan didasarkan pada skor atau nilai total masing-masing alternative.

II.3. Pengertian Basis Data (Database)

Basis data dapat dipahami sebagai suatu kumpulan data terhubung (*interrelated data*) yang disimpan secara bersama-sama pada suatu media, tanpa

mengatap satu sama lain atau tidak perlu suatu kerangkapan data (kalaupun ada maka kerangkapan data tersebut harus seminimal mungkin dan terkontrol [*controlled redudancy*]), 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 (Edy Sutanta, 2011 : 29-30).

Berdasarkan tingkat kompleksitas nilai data, tingkatan data dapat disusun dalam sebuah hierarki, mulai dari yang paling sederhana hingga paling sederhana hingga paling kompleks (Edy Sutanta, 2011 : 35-36).

1. Sisem basis data, merupakan 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.
2. Basis data, merupakan sekumpulan dari bermacam-macam tipe *record* yang memiliki hubungan antar-*record* dan rincian data terhadap obyek tertentu.
3. File, merupakan sekumpulan *record* sejenis secara relasi yang tersimpan dalam media penyimpanan sekunder.
4. *Record,* merupakan *field/*atribut/data item yang saling berhubungan terhadap obyek tertentu.

5. *Data item/field/*atribut, merupakan unit terkecil yang disebut data, sekumpulan *byte* yang mempunyai makna.
6. *Data agregate*, merupakan sekumpulan data item/*field*/atribut dengan ciri tertentu dan diberi nama.
7. *Byte*, adalah bagian terkecil yang dialamatkan dalam memori. *Byte* merupakan sekumpulan *bit* yang secara konvensional terdiri atas kombinasi 8 *bit* biner yang menyatakan sebuah karakter dalam memori (1 *byte* = 1 karakter).
8. *Bit*, adalah sistem biner yang terdiri atas dua macam nilai, yaitu 0 dan 1. Sistem biner merupakan dasar yang dapat digunakan untuk komunikasi antara manusia dan mesin (komputer).

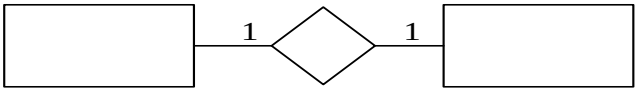
II.4. *Entity Relationship Diagram* (ERD)

ERD merupakan suatu model untuk menjelaskan hubungan antar data dalam basis data berdasarkan obyek-obyek dasar data yang mempunyai hubungan antar relasi.ERD untuk memodelkan struktur data dan hubungan antar data dengan notasi dan simbol.

“Entity Relational Diagram berfungsi untuk memodelkan struktur data dan hubungan dengan antar data, serta untuk menguji model dengan mengabaikan proses yang harus dilakukan ERD menggunakan sejumlah simbol untuk menggambarkan struktur data hubungan antar data” (Jurnal Ekonomi dan Teknik Informatika ; Heri Sudibyo : 2014 ; 71).

a. Relasi 1-1 (one to one relationship)

Satu entity anggota gugus diasosiasikan dengan tepat satu entity anggota gugus yang lain.

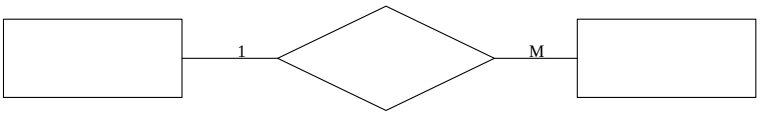


Gambar II.1. *One to One Relationship*

(Sumber : Tata Sumitra, 2014 ; 18)

Relasi 1-banyak (one to many relationship)

Suatu entity anggota gugus diasosiasikan dengan satu atau lebih entity anggota gugus yang lain. Sebaliknya satu entity anggota gugus yang lain tersebut diasosiasikan dengan tepat satu entity anggota gugus pasangannya.

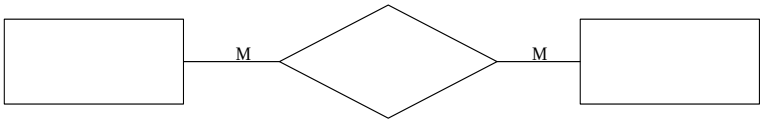


Gambar II.2. *One to Many Relationship*

(Sumber : Tata Sumitra, 2014 ; 18)

b. Entity Relationship Diagram (ERD)

Entity Relation Diagaram (ERD) adalah gambaran atau ilustrasi yang menjelaskan secara singkat antara media penyimpanan data yang satu ke data yang lain dan antara data store dengan atribut yang mengandung himpunan obyek yang bersifat unik untuk menghubungkandata. Kumpulan-kumpulan data entity relationship, atribut dan pemetaan batasan-batasan.



Gambar II.3. ***Many to Many Relationship***
(Sumber : Tata Sumitra, 2014 ; 18)

II.5. Kamus Data

Kamus Data adalah katalog fakta tentang data dan kebutuhan-kebutuhan informasidari suatu sistem informasi agar pengguna sistem dapat mendefinisikan data yang mengalirdi sistem dengan lengkap (Jurnal Ekonomi dan Teknik Informatika ;Heri Sudibyo : 2014 ; 71).

Kamus Data adalah catalog fakta tentang data dan kebutuhan informasi dari suatu sistem informasi. Dengan menggunakan kamus data, analisis system dapat mendefinisikan data yang mengalir pada system dengan lengkap, kamus data dapat dibuatpada tahap analisis system maupun pada tahapperancangan system (Jurnal Ilmu Komputer ;Tata Sumitra : 2014 ; 19).

II.6. Normalisasi

Menurut MartinNormalisasi diartikan sebagai suatu teknik yangmenstrukurkan/mendekomposisi data dalam cara-cara tertentu untuk mencegah timbulnya permasalahan pengolahan data dalam basis data. Permasalahan yang dimaksud adalah berkaitan dengan penyimpangan-penyimpangan *(anomalies)* yang terjadi akibat adanya kerangkapan data dalam relasi dan in-efisiensi pengolahan (Edy Sutanta : 2011 ; 174).

Proses normalisasi menghasilkan relasi yang optimal, yaitu(Edy Sutanta ; 2011 : 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.(Edy Sutanta ; 2011 : 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.7. *SQL Server 2008*

Microsoft SQL Server adalah salah aplikasi DBMS yang sudah sangat banyak digunakan oleh para pemrogram aplikasi basis data. Contoh DBMS lainnya adalah: MySQL (freeware), PostgreeSQL (freeware), MS Access dari Microsoft, DB_2 dari IBM, Oracle dan Oracle Corp, Dbase, FoxPro, dsb (Priyanto Hidayatullah : 2012 ; 176).

SQL Server adalah sebuah *database*relasional yang dirancang untuk mendukung aplikasi dengan arsitektur *client/server* dimana *database* terdapat pada komputer pusat yang disebut *server,* dan informasi digunakan bersama–sama oleh beberapa *user* yang menjalankan aplikasi didalam komputer lokalnya yang disebut dengan *client.*Arsitektur semacam ini memberikan integritas data yang tinggi karena semua *user* bekerja dengan informasi yang sama. Melalui aturan-aturan bisnis, kendali diterapkan kepada semua *user* mengenai informasi yang ditambahkan ke dalam *database*.*Database SQL Server* dibagi kedalam beberapa komponen logikal, seperti misalnya tabel, view, dan elemen–elemen lain yang terlihat oleh *user*(Jurnal Ilmiah Mikrotek ;Rika Yunitarini : 2013 ; 5).

II.8. *Microsoft Visual Basic* 2010

Pemrograman Visual Basic Net adalah bahasa pemrograman terpopuler.Ini merupakan pemrograman yang berjalan diatas platform NET Framework.Karena

itu setiap kali pemrograman VB.NET ini merilis versi barunya, tentu saja akan diikuti atau berbarengan dengan perkembangan Net Framwork terbaru (Edy Winarno, et al., : 2013 ; 141).

Microsoft Visual Basic.NET (VB.NET) adalah suatu pengembangan aplikasi bahasa pemrograman berbasis Visual Basic dan merupakan bahasa pemrograman terbaru buatan Microsoft setelah Microsoft Visual Basic 6.0. Pengembangan yang signifikan dari VB.NET ialah kemampuannya memanfaatkan platform NET, sehingga pengguna dapat membuat aplikasi Windows, aplikasi konsol, pustaka kelas, layanan NT, aplikasi web form, dan XML Web Service, yang secara keseluruhan memungkinkan integrasi tanpa batas dengan bahasa pemrograman lain sehingga berpeluang untuk berintegrasi dengan web (Journal Speed – Sentra Penelitian Engineering dan Edukasi ; Widiana Mulyani : 2015 ; 16).

Pada akhir tahun 1999, Teknologi .NETdiumumkan. Microsoft memosisikan teknologi tersebut sebagai *platform* untuk membangun XML Web *Services*. XML Web *services* memungkinkan aplikasi tipe manapun dan dapat mengambil data yang tersimpan pada server dengan tipe apapun melalui internet.

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. (Priyanto Hidayatullah : 2012 ; 5).

II.9. *Unified Modeling Language* (UML)

UML (*unified modeling language*) adalah bahasa pemodelan untuk sistem atau perangkat lunak yang berparadigma 'Berorientasi Objek'.Pemodelan (*Modeling*) digunakan untuk penyederhanaan permasalahan-permasalahan yang kompleks sedemikian rupa sehingga lebih mudah dipelajari dan dipahami. Tujuan pemodelan adalah sebagai sarana analis, pemahaman visualisasi, dan komunikasi antar anggota tim pengembang (saat seorang analis perangkat lunak bekerja dalam tim yang beranggotakan beberapa/banyak anggota), serta sebagai sarana dokumentasi (yang bermanfaat untuk melakukan pengujian terhadap perangkat lunak yang telah diselesaikan).

UML yang sesungguhnya merupakan metodologi kolaborasi antara metode-metode Booch yang dikembangkan oleh Graddy Booch. OMT (*Object Modeling Technique*) yang dikembangkan oleh DR. James Rumbaugh, serta OOSE (*Object Oriented Softwere Engineering*) yang dikembangkan oleh Ivar Jacobson, dan beberapa metode lainnya, merupakan metodelogi yang paling sering/paling tepat digunakan saat ini yang mengadaptasi penggunaan bahasa-bahasa pemograman yang berparadigma berorientasi objek (Jurnal Online ICT STMIK IKMI ; Achmad Hamzah Nasrullah et al., : 2015 ; 6).

II.9.1. Tujuan Pemanfaatan *Unified Modeling Language* (UML)

Tujuan dari penggunaan diagram seperti diungkapan oleh Schmuller J. (2004), *"The purpose of the diagrams is to present multiple views of a system; this set of multiple views is called a model".*

Berikut tujuan utama dalam desain UML adalah (Sugrue J. 2009) : (Jurnal Informatika Mulawarman ; Haviluddin : 2011 ; 2).

1. Menyediakan bagi pengguna (analisis dan desain sistem) suatu bahasa pemodelan visual yang ekspresif sehingga mereka dapat mengembangkan dan melakukan pertukaran model data yang bermakna.
2. Menyediakan mekanisme yang spesialisasi untuk memperluas konsep inti.
3. Karena merupakan bahasa pemodelan visual dalam proses pembangunannya maka UML bersifat independen terhadap bahasa pemrograman tertentu.
4. Memberikan dasar formal untuk pemahaman bahasa pemodelan.
5. Mendorong pertumbuhan pasar terhadap penggunaan alat desain sistem yang berorientasi objek (OO).
6. Mendukung konsep pembangunan tingkat yang lebih tinggi seperti kolaborasi, kerangka, pola dan komponen terhadap suatu sistem.
7. Memiliki integrasi praktik terbaik.

Ada 4 (empat) prinsip dasar dari pemrograman berorientasi obyek yang menjadi dasar kemunculan UML, yaitu *abstraksi, enkapsulasi, modularitas* dan *hirarki*.Berikut dijelaskan satu persatu secara singkat.

1. *Abstraksi* memfokuskan perhatian pada karakteristik obyek yang paling penting dan paling dominan yang bisa digunakan untuk membedakan obyek tersebut dari obyek lainnya.

2. *Enkapsulasi* menyembunyikan banyak hal yang terdapat dalam obyek yang tidak perlu diketahui oleh obyek lain. Dalam praktek pemrograman, enkapsulasi diwujudkan dengan membuat suatu kelas interface yang akan dipanggil oleh obyek lain, sementara didalam obyek yang dipanggil terdapat kelas lain yang mengimplementasikan apa yang terdapat dalam kelas *interface*.
3. *Modularitas* membagi sistem yang rumit menjadi bagian-bagian yang lebih kecil yang bisa mempermudah developer memahami dan mengelola obyek tersebut.
4. *Hirarki* berhubungan dengan abstraksi dan modularitas, yaitu pembagian berdasarkan urutan dan pengelompokkan tertentu. Misalnya untuk menentukan obyek mana yang berada pada kelompok yang sama, obyek mana yang merupakan komponen dari obyek yang memiliki hirarki lebih tinggi. Semakin rendah hirarki obyek berarti semakin jauh abstraksi dilakukan terhadap suatu obyek.

II.9.2. Komponen-komponen *Unified Modeling Language* (UML)

Sejauh ini para pakar merasa lebih mudah dalam menganalisa dan mendesain atau memodelkan suatu sistem karena UML memiliki seperangkat aturan dan notasi dalam bentuk grafis yang cukup spesifik (Sugrue J. 2009).

Komponen atau notasi UML diturunkan dari 3 (tiga) notasi yang telah ada sebelumnya yaitu *Grady Booch, OOD (Object-Oriented Design), Jim Rumbaugh,*

OMT (Object Modelling Technique), dan *Ivar Jacobson OOSE (Object-Oriented Software Engineering).*

Pada UML versi 2 terdiri atas tiga kategori, diantaranya (Jurnal Informatika Mulawarman ; Haviluddin : 2011 ; 3)

1. Struktur Diagram

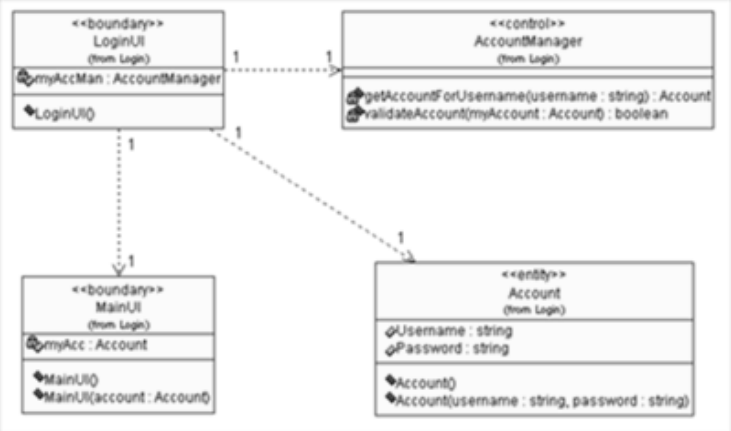
   Menggambarkan elemen dari spesifikasi dimulai dengan kelas, obyek, dan hubungan mereka, dan beralih ke dokumen arsitektur logis dari suatu sistem. Beberapa struktur diagram dalam UML terdiri atas :

   a. *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.

      Class memiliki tiga area pokok :

      1) Nama (dan *stereotype*)
      2) Atribut
      3) Metoda



Gambar II.4. Notasi Class Diagram

(Sumber : Haviluddin, 2011 : 3)

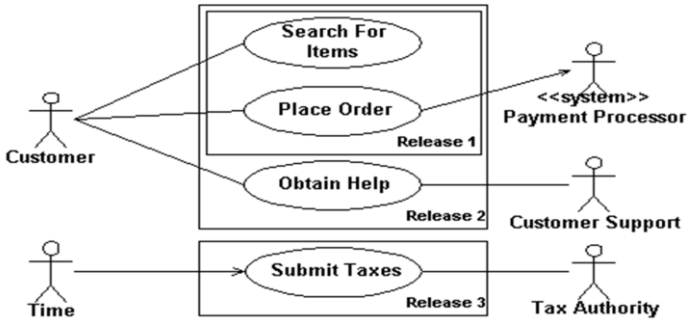
2. *Behavior Diagram*

   Menggambarkan ciri-ciri behavior/metode/fungsi dari sebuah sistem atau *business process*.*Behavior diagram* dalam UML diantaranya terdiri atas :

   a. *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*. *Use Case* memiliki dua istilah, yaitu :

      1) *System use case*; interaksi dengan sistem.
      2) *Business use case*; interaksi bisnis dengan konsumen atau kejadian nyata

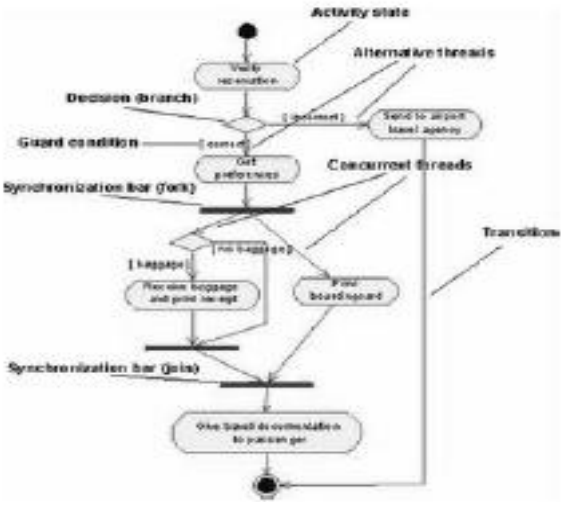


Gambar II.5. Notasi Use Case Diagram

(Sumber : Haviluddin, 2011 : 4)

b. *Activity diagram*

Menggambarkan aktifitas-aktifitas, objek, *state*, *transisi state* dan *event*. Dengan kata lain kegiatan diagram alur kerja menggambarkan perilaku sistem untuk aktivitas. Berikut notasi *object diagram* dapat dilihat pada Gambar II.6.di bawah ini.



Gambar II.6. Notasi *Activity* Diagram

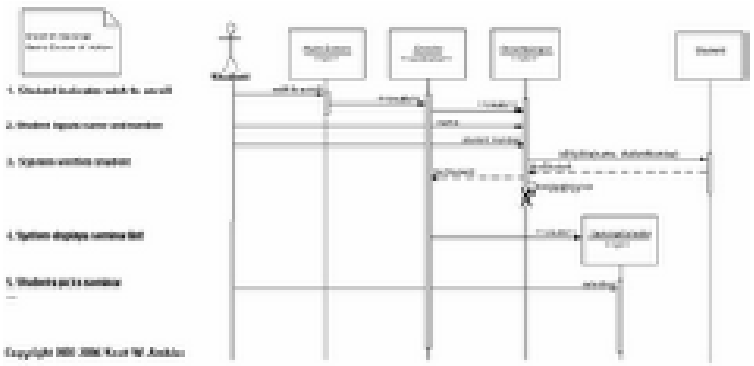
(Sumber : Haviluddin, 2011 : 4)

3. *Interaction Diagram*

Bagian dari *behavior diagram* yang menggambarkan interaksi objek. *Interaction diagram* dalam UML salah satunya adalah :

a. *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.*



Gambar II.7. Notasi *Sequence* Diagram
(Sumber : Haviluddin, 2011 : 5)

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).*