BAB II

LANDASAN TEORI

II.1. Sistem Pendukung Keputusan

Sistem pendukung keputusan menurut *Gorry Dan Scout Morton* adalah sistem berbasis komputer interaktif, yang membantu para pengambil keputusan untuk menggunakan data dan berbagai model untuk memecahkan masalah-masalah tidak terstruktur, Perkembangan teknologi informasi telah memungkinkan pengambilan keputusan dapat dilakukan dengan lebih cepat dan cermat (Nila Susanti :2013:329).

Sistem Pendukung keputusan atau *Decision Support System* (DSS) biasanya dibangun untuk mendukung solusi atas suatu masalah atau untuk suatu peluang. Aplikasi Sistem Pendukung Keputusan (SPK) digunakan dalam Pengambilan keputusan. Aplikasi sistem pendukung keputusan menggunakan CBIS (*Computer Based Information System*) yang fleksibel, interaktif, dan dapat diadaptasi, yang dikembangkan untuk mendukung solusi atas masalah manajemen spesifikasi yang tidak terstruktur (Nila Susanti :2013:329).

II.1.1 Komponen-Komponen Sistem Pendukung Keputusan

Sistem pendukung keputusan terdiri dari 4 komponen utama, yaitu :

1. Subsistem manajemen data berfungsi sebagai memasukkan suatu databaseyang berisi data yang relevan untuk situasi dan dikelola oleh perangkat lunak yang disebut sistem manajemen database (DBMS).

Knowledge Baseberisi semua fakta, ide, hubungan dan interakasi suatu domain tertentu.

2. Subsistem manajemen basis pengetahuan bertugas untuk mendukung semua subsistem lain atau bertindak sebagai suatu komponen independen. Memberikan intelegensi untuk memperbesar pengetahuan pengambil keputusan.
3. Subsistem manajemen model Merupakan paket perangkat lunak yang memasukkan model keuangan statistik, ilmu manajemen atau model kuantitatif lainnya yang memberikan kapabilitas analitik dan manajemen perangkat lunak yang tepat.
4. Subsistem antar muka pengguna (dialog) untuk mengimplementasikan sistem kedalam program aplikasi sehingga pengguna atau pemakai dapat berkomunikasi dengan sistem yang dirancang.

II.1.2 Karakteristik Sistem Pendukung Keputusan

Menurut Novriansyah (2015:1) **k**arakteristik sistem pendukung keputusan adalah sebagai berikut:

1. Mendukung proses pengambilan keputusan suatu organisasi atau perusahaan.
2. Adanya *interface* manusia/mesin dimana manusia (*user*) tetap memegang kontrol proses pengambilan keputusan
3. Mendukung pengambilan keputusan untuk membahas masalah terstruktur, semi terstruktur serta mendukung beberapa keputusan yang saling berinteraksi.

4. Memiliki kapasitas dialog untuk memperoleh informasi sesuai dengan kebutuhan.
5. Memitiki sub sistem yang terintegrasi sedemikian rupa sehingga dapat berfungsi sebagai kesatuan sistem.
6. Memiliki dua komponen utama yaitu data dan model.

II.1.3 Kriteria Sistem Pendukung Keputusan

Menurut Novriansyah (2015:2) kriteria atau ciri-ciri sistem pendukung keputusan adalah sebagai berikut:

1. Banyak pilihan / alternatif.
2. Ada kendala / surat.
3. Mengikuti suatu pola/ model
4. Banyak input / variabel
5. Ada faktor resiko. Dibutuhkan kecepatan, ketepatan dan keakuratan.

II.1.4 Fase Dalam Proses Pengambilan Keputusan

Menurut Novriansyah (2015:2) ada 3 (tiga) fase dalam proses pengambilan keputusan diantaranya sebagai berikut :

1. *Intellegence*

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

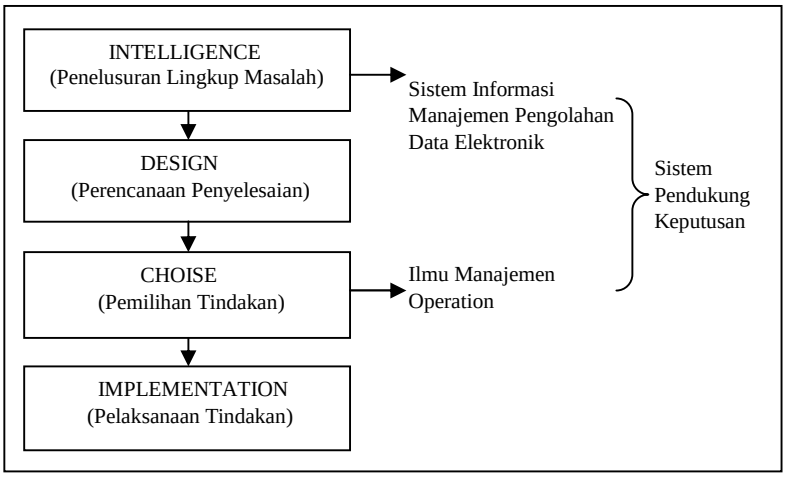
2. *Design*

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

3. *Choice*

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

   Fase dalam proses pengambilan keputusan dapat dilihat pada Gambar II.1 seperti berikut.



Gambar II.1 Fase Proses Pengabilan Keputusan
(*Sumber : Novriansyah ; 2016*)

II.1.5 Tujuan Sistem Pendukung Keputusan

Menurut Novriansyah (2015:4) tujuan dari sistem pendukung keputusan adalah pengambilan keputusan diantaranya sebagai berikut :

1. Membantu dalam pengambilan keputusan atas masalah yang terstruktur.

2. Memberikan dukungan atas pertimbangan manajer dan bukannya dimaksudkan untuk menggantikan fungsi manajer.
3. Meningkatkan efektifitas keputusan yang diambil lebih dari pada perbaikan efesiensinya.
4. Kecepatan komputasi komputer memungkinkan para pengambil keputusan untuk banyak melakukan komputasi secara cepat dengan biaya yang rendah.
5. Peningkatan produktivitas membangun suatu kelompok pengambilan keputusan, terutama para pakar, biasa sangat mahal.

II.2. Penilaian Karyawan

Karyawan adalah orang yang menjual jasanya kepada orang lain atau orang yang bekerja pada sebuah perusahaan/lembaga/instansi. (Jannah, 2015:2). Penilaian karyawan, pada dasarnya adalah upaya penilaian terhadap kinerja karyawan. Secara umum dapat diartikan sebagai upaya guna mengadakan pengukuran atas kinerja dari setiap karyawan perusahaan. Hal ini dikaitkan dengan tingkat produktivitas dan efektivitas kerja dar karyawan tersebut dalam menghasilkan karya tertentu, sesuai dengan deskripsi tugas yang diberikan perusahaan kepada karyawan yang bersangkutan. (Budiharjo, 2014:13)

Hasil dari pengukuran kinerja karyawan atau hasil dari penilaian karyawan ini secara umum akan digunakan sebagai bahan pertimbangan dalam upaya peningkatan produktivitas dan efektivitas perusahaan, yang dilakukan secara terus-menerus, berlanjut, dan berkesinambungan.

Secara umum. penilaian terhadap karyawan, pegawai, ataupun staf suatu organisasi memiliki berbagai manfaat, baik bagi organisasi maupun bagi karyawan itu sendiri. Bagi karyawan, akan menyebabkan terpicunya semangat berkompetisi untuk menjadi lebih baik ke depannya. Salah satunya ditandai dengan peningkatan etos kerja para karyawan itu sendiri. Sementara itu, bagi organisasi akan berdampak pada adanya peningkatan produktivitas organisasi.

II.2.1 Bonus

Jenis kompensasi finalsial lain yang ditetapkan perusahaan adalah berupa pemberian bonus. Pemberian bonus kepada karyawan ini dimaksudkan untuk meningkatkan produktifitas kerja dan semangat kerja karyawan. Pengertian bonus menurut simamora adalah pembayaran sekaligus yang diberikan karena memenuhi sasaran kinerja.

a. Uang dibayar sebagai balas atas hasil pekerjaan yang
b. Telah dilaksanakan apabila melebihi target diberikan secara sekali terima tanpa sesuatu ikatan di masa yang akan datang
c. Beberapa persen dari laba yang kemudian dibagikan kepada yang berhak menerima bonus.

Bonus diberikan apabila karyawan mempunyai profitabilitas atau keuntungan dari seluruh penjualan tahun lalu. Penentuan besarnya pemberian bonus adalah berdasarkan kebijakan perusahan, tidak ada ketetapan yang pasti mengenai bonus yang diberikan. Menyatakan bahwa tidak ada aturan yang pasti mengenai sistem

perhitungan bonus dan beberapa perusahan tidak memiliki formula untuk mengembangkan dana bonus.

II.3. *Multi Attributte Utility Theory* (MAUT)

Multi Attribute Utility Theory (MAUT) merupakan suatu skema yang evaluasi akhir, *v(x),* dari suatu objek *x* didefinisikan sebagai bobot yang dijumlahkan dengan suatu nilai yang relevan terhadap nilai dimensinya. Ungkapan yang biasa digunakan untuk menyebutnya adalah nilai utilitas (Jannah, 2015:2).

Multi Attribute Utility Theory digunakan untuk merubah dari beberapa kepentingan kedalam nilai numerik dengan skala *0-1* dengan *0* mewakili pilihan terburuk dan *1* terbaik. Hal ini memungkinkan perbandingan langsung beragam ukuran. Hasil akhirnya adalah urutan peringkat dari evaluasi alternatif yang menggambarkan pilihan dari para pembuat keputusan. Nilai evaluasi seluruhnya dapat didefinisikan dengan persamaan:

$$v(x) = \sum_{i=1}^{n} w_j x_{ij} \quad \text{…….....…..} \quad (1)$$

Dimana $v_i\ (x)$ merupakan nilai evaluasi dari sebuah objek ke *i* dan w_i merupakan bobot yang menentukan nilai dari seberapa penting elemen ke *i* terhadap elemen lainnya. Sedangkan n merupakan jumlah elemen. Total dari bobot adalah *1*.

$$\sum_{i=1}^{n} w_i = 1 \quad \text{…….....…..} \quad (2)$$

Untuk setiap dimensi, nilai *evaluation* v_i *(x)* didefinisikan sebagai penjumlahan dari atribut-atribut yang relevan.

$$v_i(x) = \sum_{a \in A} w_{ai} . v_{ai}(I(a)) \quad \text{…….....……..} \quad (3)$$

Langkah-langkah dalam metode MAUT adalah sebagai berikut :

1. Pecah sebuah keputusan ke dalam dimensi yang berbeda.
2. Tentukan bobot relatif pada masing-masing dimensi.
3. Daftar semua alternatif.
4. Masukkan *utility* untuk masing-masing alternative sesuai atributnya.
5. Kalikan *utility* dengan bobot untuk menemukan nilai dari alternatif.

Multi Attribute Utility Theory (MAUT) merupakan merupakan metode untuk membantu pengambilan sejumlah alternatif keputusan. Untuk i alternatif keputusan dengan j atribut, additive utility model dinyatakan dengan rumus berikut (Aries Susanty :2015).

$$u(x_i) = \sum_{r=1}^{n} k_j * u_j\left(x_{ij}\right) \quad \text{.....} \quad (4)$$

dimana

$$\sum_{j=1}^{n} k_j = 1 \quad \text{.....} \quad (5)$$

kj = bobot relatif dari atribut ke j

uj (xij) = utilitas dari setiap outcome xij untuk setiap atribut j

Langkah-langkah yang dilakukan dalam metode MAUT dapat dituliskan sebagai berikut:

1. Pecahkan atau uraikan sebuah keputusan dalam atribut yang berbeda (biaya, waktu, infrasturktur, dan pendapat pengusaha atau para ahli dibidangnya).
2. Tentukan bobot relatif pada masing-masing atribut.
3. Daftar semua alternatif yang ada.
4. Tentukan nilai indeks dari setiap atribut dan tentukan nilai utilitas dari setiap atribut dengan menggunakan rumus (3) sampai dengan rumus (7).
5. Masukan nilai utilitas untuk masing-masing alternatif sesuai atributny dan
6. Kalikan nilai utilitas dengan bobot untuk menentukan nilai masing-masing alternatif.

II.4. *Java*

Java merupakan bahasa pemrograman tingkat tinggi yang memiliki karakteristik *simple*, *object oriented*, *distributes*, *interpreted* dan memiliki performa yang tinggi. (Andi Publisher, 2015:2).

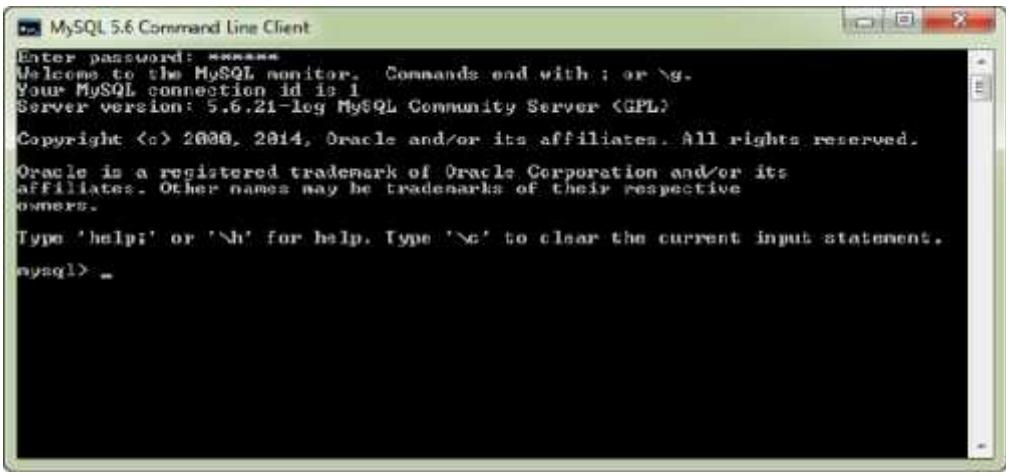
Bahasa pemrograman java merupakan *compiler* sekaligus *interpreter*, dimana sebagai *compiler* program yang telah dibuat akan diubah menjadi *Java Bytecodes*. Sedangkan sebagai *interpreter*, *Java Bytecodes* tersebut dijalankan pada komputer. Pada bahasa pemrograman *Java* terdapat berbagai macam fitur yang disediakan oleh platform teknologi Java, yaitu *Java Virtual Machine* (*JVM*), *Garbage Collection* dan *Code Security*.

Gambar II.2 Tampilan Java
(*Sumber : Andi Publisher ; 2015)*

II.5 *MySQL*

MySQL adalah adalah salah satu jenis *database server* yang sangat terkenal di dunia. *MySQL* termasuk jenis *RDBMS* (*Relational Database Managemen System*). Oleh karena itu, istilah seperti tabel, baris dan kolom banyak digunakan pada *MySQL*. (Andi Publisher, 2015:6).

Bahasa yang digunakan oleh *MySQL* tentu saja adalah SQL-*standar* bahasa basis data relasional di seluruh dunia saat ini. Tampilan awal MySQL dapat dilihat pada gambar II.3 seperti berikut.



Gambar II.3 Tampilan Awal MySQL
(*Sumber : Andi Publisher ; 2015*)

II.6. *UML* (*Unified Modelling Language*)

UML merupakan kependekan dari *Unified Modeling Language* yaitu diagram dan metode standar untuk memodelkan dan merepresentasikan *object oriented software* dan sistem bisnis. (Mulyani, 2016:243).

Beberapa fungsi dan kegunaan dari *UML* yaitu (Mulyani, 2016:244) :

1. *Visualizing,* yaitu sebagai alat komunikasi konseptual model antara tim pengembang sistem (sistem analis dengan programmer)
2. *Specifying,* yaitu sebagai *tools* yang digunakan untuk memodelkan sistem secara tepat dan jelas.
3. *Constructing,* yaitu *UML* sebagai bahasa gratis mampu melakukan *mapping* dan konseptual model kedalam bahasa pemrograman.
4. *Documenting,* yaitu *UML* digunakan sebagai *tools* untuk melakkukan dokumentasi teknis sebuah sistem.

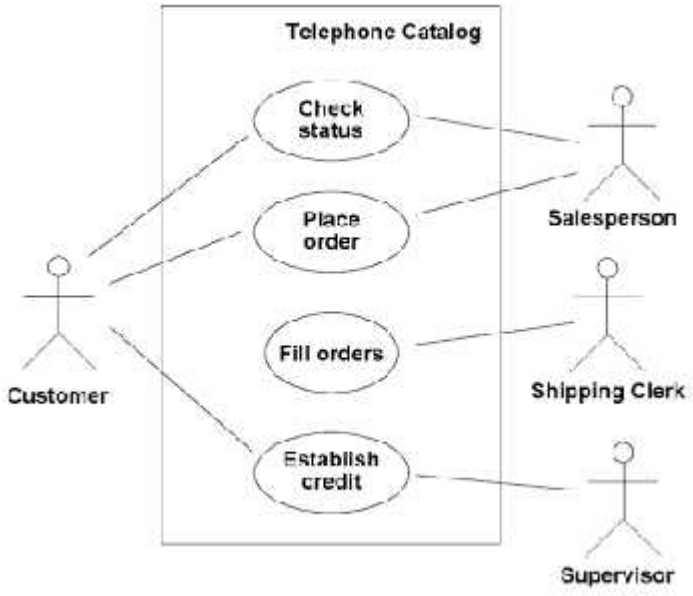
Diagram-diagram yang terdapat dalam *UML* sangat banyak, berikut ini beberapa diagram yang sering digunakan dalam pengembangan sistem yaitu :

1. Use Case Model

   Use case model merupakan kumpulan diagram dan text yang saling bekeria sama untuk mendokumentasikan bagaimana user (aktor) berinteraksi dengan sistem. Use case model terdiri dari beberapa diagram :

   1) *Use case diagram* yaitu diagram yang menggambarkan dan merepresentasikan aktor, *use cases*, dan *dependencies* suatu proyek dimana tujuan dan diagram ini adalah untuk menjelaskan konsep

hubungan antara sistem dengan dunia luar. *Use case diagram* dapat dilihat pada gambar II.4 seperti berikut.

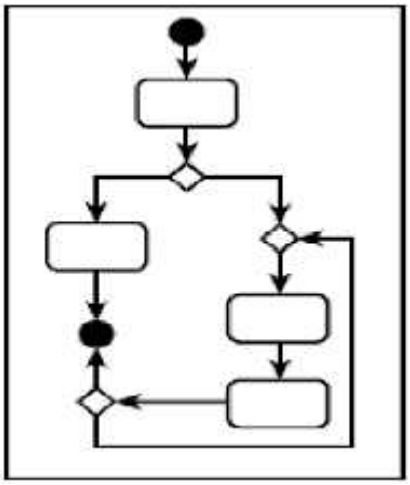


Gambar II.4 ***Use Case Diagram***
(*Sumber : Sri Mulyani ; 2016*)

2) *Use case narrative* yaitu deskripsi yang menjelaskan use case diagram. Pada *use case diagram* sistem hanya digambarkan secara sederhana menggunakan simbol *use case* yang berhubungan (*relationship*) dengan aktor, sehingga terkadang diperlukan deskripsi yang menjelaskan dan proses tersebut. *Use case narrative* dapat dilihat pada gambar II.5 seperti berikut.

Gambar II.5 ***Use Case Narrative***
(*Sumber : Sri Mulyani ; 2016*)

3) *Use case scenario* yaitu pemecahan kemungkinan logika pada *use case diagram*. *Use case scenario* dapat dilihat pada gambar II.6 seperti berikut.

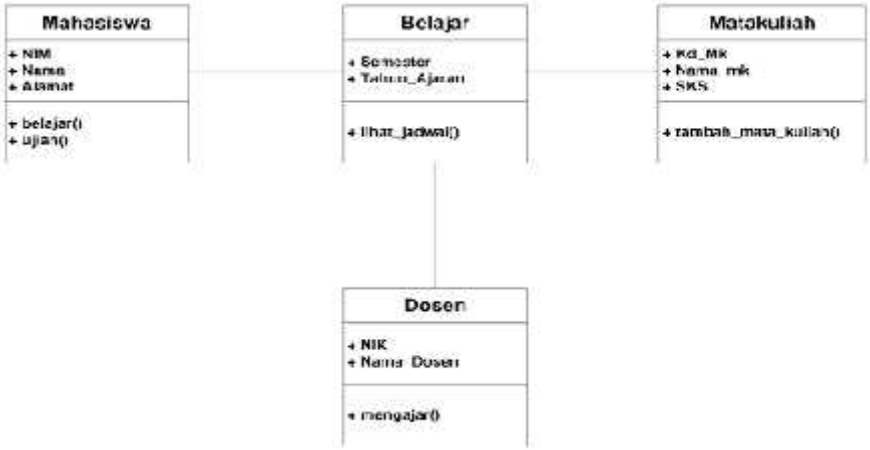


Gambar II.6 ***Use Case Scendario***
(*Sumber : Sri Mulyani ; 2016*)

2. Class Diagram

Class diagram adalah diagram yang digunakan untuk merepresentasikan kelas, komponen-komponen kelas dan hubungan anatara masing-masing

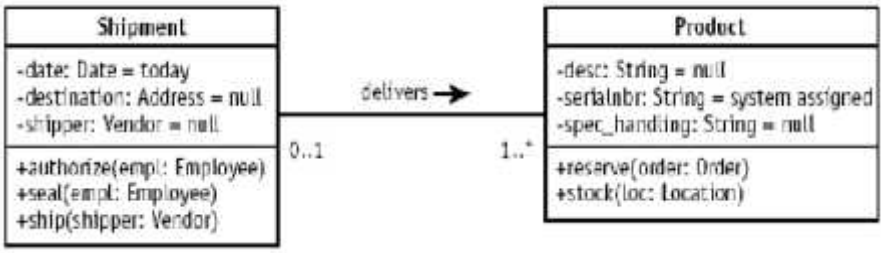
kelas. Selain itu class diagram mendeskripsikan jenis-jenis objek dalam sistem dan berbagai macam hubungan statis yang terdapat diantara mereka. *Class diagram* juga menunjukan *property* dan operasi sebuah kelas serta batasan-batasan yang terdapat dalam hubungan-hubungan objek tersebut. *UML* menggunakan istilah fitur sebagai istilah umum yang meliputi property dan operasi sebuah kelas. *Class diagram* dapat dilihat pada gambar II.7 seperti berikut.



Gambar II.7 *Class Diagram*
(*Sumber : Sri Mulyani ; 2016*)

3. *Object Diagram*

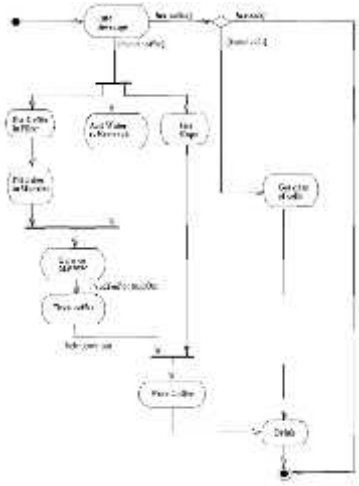
Object Diagram adalah diagram yang digunakan untuk menggambarkan dan merepresentasikan objek dan hubungan antar objek tersebut, selain itu *object diagram* juga dapat digunakan sebagai diagram untuk menunjukan sebuah konfigurasi contoh dan sebuah objek. *Object diagram* dapat dilihat pada gambar II.8 seperti berikut.



Gambar II.8 ***Object Diagram***
(*Sumber : Sri Mulyani ; 2016*)

4. Activity Diagram

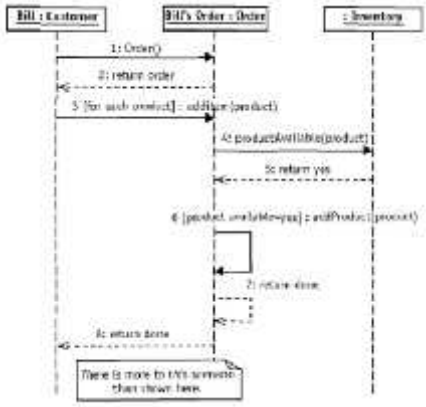
Jika sebelumnya Anda sudah pernah menggunakan *flowchart*, maka Anda akan mudah dalam memahami dan mempelajari *activity diagram*, karena *diagram* ini sangat mirip sekali dengan *fiowchart*, perbedaanya adalah *activity diagram* memiliki kemampuan untuk melakukan percabangan aktivitas, selain itu *activity* diagram juga memungkinkan pemisahan aktivitas antar aktor. *Activity diagram* adalah diagram *UML* yang digunakan untuk menggambarkan alur aktivitas dari satu proses. *Activity diagram* memungkinkan siapapun yang melakukan proses untuk memilih urutan dalam melakukannya, dengan kata lain diagram hanya menyebutkan aturan-aturan rangkaian dasar yang harus kita ikuti. Hal ini penting untuk pemodelan bisnis karena proses-proses sering muncul secara parallel. Ini juga berguna pada algoritma yang bersamaan, dimana urutan-urutan independen dapat melakukan hal-hal secara parallel. *Acitvity diagram* dapat dilihat pada gambar II.9 seperti berikut.



Gambar II.9 ***Activity Diagram***
(*Sumber : Sri Mulyani ; 2016*)

5. Sequence Diagram

Sequence diagram adalah diagram yang menggambarkan interaksi antar objek. *Sequence diagram* secara khusus menjabarkan *behavior* sebuah skenario tunggal. Diagram tersebut menunjukan sejumlah objek contoh dan pesan-pesan yang melewati objek ini dalam sebuah *use case*. *Seqquence diagram* dapat dilihat pada gambar II.10 seperti berikut.



Gambar II.10 ***Sequence Diagram***
(*Sumber : Sri Mulyani ; 2016*)

II.7 *Normalisasi* (*Normalization*)

Normalisasi adalah salah satu cara untuk meminimalisir pengulangan data (*data redundancy*), normalisasi akan diperlukan jika ada indikasi bahwa tabel yang kita buat tidak baik (terjadi pengulangan informasi, potensi inkonsistensi data pada operasi pengubahan, tersembunyinya informasi tertentu dan lain sebagainya) dan diperlukan supaya jika tabel-tabel yang didekomposisi kita gabung kembali dapat menghasilkan tabel awal sebelum didekomposisi, sehingga diperoleh tabel yang baik. Hasil dari normalisasi adalah himpunan-himpunan data (*tabel-tabel*) dalam bentuk normal (*normal form*). (Mulyani, 2016:132)

Beberapa kegunaan dari penggunaan normalisasi dalam perancangan database adalah (Mulyani, 2016:132).

1. Meminimalisir pengulangan data (*data redundancy*)
2. Memudahkan identifikasi *entity* objek.

Beberapa bentuk normal yaitu (Mulyani, 2016:132):

1. Bentuk Normal I (*First Normal Form / 1-NF*)

   Suatu relasi memenhi *1-NF* jika dan hanya jika setiap *attribute* dan relasi tersebut hanya memiliki nilai tunggal dalam 1 (satu) baris *record* (memisahkan group berulang).

2. Bentuk Normal II (*Second Normal Form/2-NF*)

   Suatu relasi memenuhi *2-NF* jika dan hanya jika memenuhi *1-NF* dan Setiap *attribute* yang bukan kunci utama tergantung secara fungsional terhadap semua attribute kunci dan bukan hanya sebagian attribute (menghilangkan ketergantungan fungsional pada sebagian/salah satu key).

3. Bentuk Normal Ill (*Third Normal Form/3-NF*)

Suatu relasi memenuhi *3-NF* jika dan hanya jika memenuhi *2-NF* dan setiap *attribute* bukan kunci tidak tergantung secara fungsional kepada *attribute* bukan kunci yang lain dalam relasi tersebut (menghilangkan ketergantungan transitif pada yang bukan *key*).

4. *Boyce-Codd Normal Form* (BCNF)

   Suatu relasi memenhi BCNF jika untuk setiap *functional dependency (FD)* terhadap setiap atribut atau gabungan atribut dalam bentuk : *X -> Y* maka *X* adalah *super key*. Tabel tersebut harus di dekomposisi berdasarkan *FD* yang ada, sehingga *X* menjadi super key dari tabel-tabel hasil dekomposisi. Setiap tabel dalam BCNF merupakan 3NF. Akan tetapi setiap *3NF* belum tentu termasuk BCNF. Perbedaannya, untuk *functional dependency* X -> A, BCNF tidak membolehkan A sebagai bagian dari *primary key*.

5. Bentuk Normal IV (*Fourth Normal Form/4-NF*)

   Suatu relasi memenuhi *4-NF* jika dan hanya jika memenuhi *BCNF* dan tabel tersebut tidak boleh memiliki lebih dari sebuah *multivalued attribute*. Untuk setiap *multivalued attribute* (*MVD*) juga harus merupakan *functional dependencies*.

Beberapa key dalam Normalisasi (Mulyani, 2016:132):

1. *Superkey* adalah sejumlah *attribute entity* yang dapat digunakan untuk mengidentifikasi objek secara unik.
2. *Candidate key* adalah *superkey* dengan jumlah attribute minimal dan dapat berdiri sendiri.
3. *Primary key* adalah *superkey* yang dipilih oleh database administrator.

4. *Foreign key* adalah *attribute* disuatu relasi (*tabel*) yang menjadi *primary key* di relasi (*label*) lain.

II.8 *Basis Data* (*Database*)

Database adalah kumpulan dari semua data yang diperlukan oleh sistem. Dengan menggunakan *database*, beberapa apalikasi berbeda bisa saling terintegrasi. (Mulyani, 2016:148).

Database merupakan komponen terpenting dalam pembangunan sistem, karena menjadi tempat untuk menampung dan mengorganisasikan seluruh data yang ada dalam sistem, sehingga dapat dieksplorasi untuk menyusun-menyusun informasi-informasi dalam berbagai bentuk.

II.8.1 DBMS (*Database Management System)*

Database Management System adalah sistem *software* yang memungkinkan pengguna untuk mendefinisikan, membuat, memelihara, dan kontrol akses ke database. (Mulyani, 2016:170).

DBMS adalah software yang berinteraksi dengan program aplikasi dan pengguna database Biasanya *DBMS* menyediakan fasilitas sebagai berikut :

1. *DDL* (*Data Definition Language*)

   DDL memungkinkan pengguna untuk menentukan tipe data dan struktur dan kendala pada data yang akan disimpan dalam *database*.

2. *DML*(*Data Manipulation Language*)

Ini memungkinkan pengguna untuk memasukkan, update, menghapus dan mengambil data dari *database* biasanya meskipun memanipulasi data bahasa (*DML*).

3. Memberikan akses kontrol ke *database* :
    1) Keamanan sistem: yang mencegah pengguna yang tidak berhak mengakses database.
    2) Integritas system: yang menjaga konsistensi data yang tersimpan.
    3) *Concurrency control system*: yang memungkinkan berbagi akses database.
    4) Pemulihan sistem *control*: yang mengembalikan *database* ke keadaan yang konsisten sebelumnya setelah perangkat keras atau kegagalan *software*.
    5) *User*-diakses katalog, yang berisi deskripsi dari data dalam *database*.