BAB II

TINJAUAN PUSTAKA

II.1. Sistem Pendukung Keputusan

Pengambilan keputusan (*Decision Making*) adalah melakukan penilaian dan menjatuhkan pilihan. Keputusan ini diambil setelah melalui beberapa perhitungan dan pertimbangan alternatif. Decision support system atau sistem pendukung keputusan secara umum didefinisikan sebagai sebuah sistem yang mampu memberikan kemampuan baik kemampuan pemecahan masalah maupun kemampuan pengkomunikasian untuk masalah semi terstruktur. Secara khusus, Sistem pendukung keputusan didefinisikan sebagai sebuah sistem yang mendukung kerja seorang manager maupun sekelompok manager dalam memecahkan masalah semi terstruktur dengan cara memberikan informasi ataupun usulan menuju pada keputusan tertentu (Muhammad Yudin Ritonga ; 2014 : 19).

II.1.1. Komponen Pengambilan Keputusan

Sistem pendukung keputusan terdiri dari 4 komponen utama, yaitu :

1. Subsistem manajemen data berfungsi sebagai memasukkan suatu *database* yang berisi data yang relevan untuk situasi dan dikelola oleh perangkat lunak yang disebut sistem manajemen *database* (DBMS). *Knowledge Base* berisi 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. Ia 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 (Nila Susanti ; 2013 : 329).

II.2. Nama

Nama adalah sebuah pemberian yang berharga dari orangtua terhadap anaknya. Selain itu, nama merupakan sebuah identitas bagi seseorang. Tanpa adanya sebuah nama akan menjadi sulit bagi kita untuk menentukan dan memanggil seseorang. Namun untuk menentukan sebuah nama kita perlu mempertimbangkan arti tentang nama itu sendiri, karena nama bukan hanya sebatas kata, tetapi nama adalah sebuah identitas dan doa yang diberikan kepada orang yang dinamainya.

Ketika orangtua hendak memberikan dan menetapkan sebuah nama pada anak yang baru lahir, maka pada hakikatnya orangtua itu sedang mendoakan bayi tersebut agar semua kehidupan di masa depannya sesuai dengan keinginan orangtuanya. Supaya akhlak, sifat, rezeki, jodoh dan sebagainya sesuai dengan

harapan orangtuanya, maka menjadi hal yang logis jika banyak para ulama yang mengatakan bahwa nama adalah sebuah doa. Sebagian ulama juga ada yang berpendapat bahwa doa adalah bentuk *tafaul*, sehingga dengan adanya nama yang bagus bisa jadi membuat dorongan untuk menjadi pribadi yang bagus juga (Tarya Nurul Musthafa ; 2014 : 172).

II.2.1. Arti dan Makna Sebuah Nama

Untuk menentukan sebuah nama yang akan diberikan kepada seorang anak, hendaknya kita mengetahui secara mendalam tentang alasan pemberian nama tersebut. ketika kita ingin menentukan nama untuk seseorang, maka ketahuilah terlebih dahulu tujuan dan maksud dibalik nama tersebut. berbicara tentang maksud dan tujuan dari sebuah nama tentu sangat erat kaitannya dengan arti dari nama itu, sebab tidak akan sempurna tujuan dan maksud dari nama tersebut jika makna dari nama itu sendiri belum kita pahami. Dalam dunia bahasa Arab, makna sebuah nama tidak terlepas dari dua makna, yakni maka *haqiqi* (makna yang sebenarnya) dan makna *majazi* (makna kiasan). Kedua makna ini bukan saja berlaku pada bahasanya semata, melainkan berlaku pula pada makna nama seseorang.

1. Maka Haqiqi

   Makna hawiwi dalam sebuah nama adalah makna leksikal, sepenuhnya tidak meyimpang dari makna asli. Oleh karena itu, makna haqiqi disebut makna apa adanya atau makna sebenarnya.

2. Makna Majazi

   Makna majazi adalah sebuah nama yang mengandung arti kiasan. Dalam ilmu *balaghah*, mengupas tentang arti kiasan yang disebut dengan *isti'arah*, yaitu makna pinjaman (Tarya Nurul Musthafa ; 2014 : 173).

II.3. Metode *Simple Additive Weight (*SAW)

Konsep dasar metode SAW adalah mencari penjumlahan terbobot dari rating kinerja pada setiap alternatif pada semua atribut. Metode SAW membutuhkan proses normalisasi matriks keputusan (X) ke suatu skala yang dapat diperbandingkan dengan semua rating alternatif yang ada (Youllia Indrawaty ; 2011 : 33).

Salah satu metode penyelesaian masalah MADM adalah dengan menggunakan metode Simple Additive Weighting (SAW). Metode SAW sering juga dikenal istilah metode penjumlahan terbobot. Konsep dasar metode SAW adalah mencari penjumlahan terbobot dari rating kinerja pada setiap alternatif dari semua atribut. Metode SAW membutuhkan proses normalisasi matriks keputusan (X) ke suatu skala yang dapat diperbandingkan dengan semua rating alternatif yang ada.

$$r_{ij} = \begin{cases} \dfrac{x_{ij}}{\underset{i}{Max}\, x_{ij}} & jika \text{ j atribut keuntungan (benefit)} \\ \dfrac{\underset{i}{Min}\, x_{ij}}{x_{ij}} & \text{jika j atribut biaya (cost)} \end{cases} \quad \text{..... (1)}$$

Keterangan :

a. Rij = nilai rating kinerja normalisasi

b. Xij = nilai atribut yang dimiliki dari setiap kriteria

c. Max xij = nilai terbesar dari setiap kriteria

d. Min xij = nilai terkecil dari setiapkriteria

e. Benefit = nilai terbesar adalah terbaik

f. Cost = nilai terkecil adalah terbaik

Dimana rij adalah rating kinerja ternormalisasi dari alternatif Ai pada atribut Cj; i=1,2,...,m dan j=1,2,...,n. Nilai *preferensi* untuk setiap alternatif (Vi) diberikan sebagai:

$$V_i = \sum_{j=1}^{n} w_j \, r_{ij}$$ ..... (2)

Keterangan :

a. Vi = rangking untuk setiap alternatif

b. Wj = nilai bobot dari setiap kriteria

c. rij = nilai rating kinerja ternormalisasi.

Nilai Vi yang lebih besar mengindikasi bahwa alternatif Ai lebih terpilih (Destriyana Darmastuti ; 2012 : 2).

II.3.1. Langkah *Simple Additive Weight (*SAW)

Langkah-langkah dari metode SAW adalah:

1. Menentukan kriteria-kriteria yang akan dijadikan acuan dalam pengambilan keputusan, yaitu C.

2. Menentukan rating kecocokan setiap alternatif pada setiap kriteria.
3. Membuat matriks keputusan berdasarkan kriteria (C), kemudian melakukan normalisasi matriks berdasarkan persamaan yang disesuaikan dengan jenis atribut (atribut keuntungan ataupun atribut biaya) sehingga diperoleh matriks ternormalisasi R.
4. Hasil akhir diperoleh dari proses perankingan yaitu penjumlahan dari perkalian matriks ternormalisasi R dengan vector bobot sehingga diperoleh nilai terbesar yang dipilih sebagai alternatif terbaik (A) sebagai solusi (Youllia Indrawaty ; 2011 : 34).

II.4. Pengertian *Visual Basic*

Visual Basic dibuat oleh *Microsoft*, merupakan salah satu bahasa pemrograman beriorientasi objek yang mudah dipelajari. Selain menawarkan kemudahan, *Visual Basic* juga cukup andal untuk digunakan dalam pembuatan berbagai aplikasi, terutama aplikasi *database*. *Visual Basic* merupakan bahasa pemrograman *event drive*, di mana program aplikasi yang dapat berupa kejadian atau *event*, misalnya ketika *user* mengklik tombol atau menekan *enter*. Jika kita membuat aplikasi dengan *Visual Basic* maka kita akan mendapatkan *file* yang menyusun aplikasi tersebut, yaitu :

1. *File Project* (*.vbp)

   File ini merupakan kumpulan dari aplikasi yang kita buat. *File project* bisa berupa *file* *.frm, *.dsr atau *file* lainnya.

2. *File Form* (*.frm)

   File ini merupakan *file* yang berfungsi untuk menyimpan informasi tentang bentuk form maupun *interface* yang kita buat (Edy Winarno ; 2010 : 83).

II.5. Pengertian *SQL Server* 2008

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

Microsoft merilis *SQL Server* 2008 dalam beberapa versi yang disesuaikan dengan segment-segment pasar yang dituju. Versi-versi tersebut adalah sebagai berikut. Menurut cara pemrosesan data pada prosesor maka *Microsoft* mengelompokkan produk ini berdasarkan 2 jenis yaitu :

a. Versi 32-bit(x86), yang biasanya digunakan untuk komputer dengan *single* prosesor (Pentium 4) atau lebih tepatnya prosesor 32 bit dan sistem operasi Windows XP.
b. Versi 64-bit(x64), yang biasanya digunakan untuk komputer dengan lebih dari satu prosesor (Misalnya Core 2 Duo) dan system operasi 64 bit seperti Windows XP 64, Vista, dan Windows 7.

Sedangkan secara keseluruhan terdapat versiversi seperti berikut ini:

a. Versi Compact, ini adalah versi "Tipis" dari semua versi yang ada. Versi ini seperti versi desktop pada *SQL Server* 2000. Versi ini juga digunakan pada *handled drvice* seperti Pocket PC, PDA, *SmartPhone*, Tablet PC.

b. Versi Express, ini adalah versi "Ringan" dari semua versi yang ada(tetapi versi ini berbeda dengan versi *compact*) dan paling cocok untuk latihan para pengembang aplikasi. Versi ini memuat *Express Manager* standar, integrasi dengan CLR dan XML (Wenny Widya ; 2012 : 3)

Gambar II.1. Tampilan *SQL Server*
(Sumber : Wenny Widya ; 2012 : 3)

II.6. Pengertian *Database*

Secara sederhana *database* (basis data/pangkalan data) dapat diungkapkan sebagai suatu pengorganisasian data dengan bantuan komputer yang memungkinkan data dapat diakses dengan mudah dan cepat. Pengertian akses dapat mencakup pemerolehan data maupun pemanipulasian data seperti menambah serta menghapus data. Dengan memanfaatkan komputer, data dapat

disimpan dalam media pengingat yang disebut *harddisk.* Dengan menggunakan media ini, keperluan kertas untuk menyimpan data dapat dikurangi. Selain itu, data menjadi lebih cepat untuk diakses terutama jika dikemas dalam bentuk *database*.

Pengaplikasian *database* dapat kita lihat dan rasakan dalam keseharian kita. *Database* ini menjadi penting untuk mengelola data dari berbagai kegiatan. Misalnya, kita bisa menggunakan mesin ATM *(*anjungan tunai mandiri / *automatic teller machine*) bank karena banktelah mempunyai database tentang nasabah dan rekening nasabah.Kemudian data tersebut dapat diakses melalui mesin ATM ketika bertransaksi melalui ATM. Pada saat melakukan transaksi, dalam konteks *database* sebenarnya kita sudah melakukan perubahan *(update)* data pada *database*di bank.Ketika kita menyimpan alamat dan nomor telepon di HP, sebenarnya juga telah menggunakan konsep *database*. Data yang kita simpan di HP juga mempunyai struktur yang diisi melalui formulir*(form)* yang disediakan. Pengguna dimungkinkan menambahkan nomor HP, nama pemegang, bahkan kemudian dapat ditambah dengan alamat *email*, alamat *web*, nama kantor, dan sebagainya (Agustinus Mujilan ; 2012 : 23).

II.7. Teknik Normalisasi

Normalisasi adalah teknik perancangan yang banyak digunakan sebagai pemandu dalam merancang basis data relasional. Pada dasarnya, normalisasi adalah proses dua langkah yang meletakkan data dalam bentuk tabulasi dengan

menghilangkan kelompok berulang lalu menghilangkan data yang terduplikasi dari tabel rasional.

Teori normalisasi didasarkan pada konsep bentuk normal. Sebuah tabel relasional dikatakan berada pada bentuk normal tertentu jika tabel memenuhi himpunan batasan tertentu. Ada lima bentuk normal yang tekah ditemukan.

II.7.1.Bentuk-bentuk Normalisasi

1. **Bentuk normal tahap pertama (1" Normal Form)**

   Contoh yang kita gunakan di sini adalah sebuah perusahaan yang mendapatkan barang dari sejumlah pemasok. Masing-masing pemasok berada pada satu kota. Sebuah kota dapat mempunyai lebih dari satu pemasok dan masing-masing kota mempunyai kode status tersendiri.

2. **Bentuk normal tahap kedua (2^{nd} normal form)**

   Definisi bentuk normal kedua menyatakan bahwa tabel dengan kunci utama gabungan hanya dapat berada pada 1NF, tetapi tidak pada 2NF. Sebuah tabel relasional berada pada bentuk normal kedua jika dia berada pada bentuk normal kedua jika dia berada pada 1NF dan setiap kolom bukan kunci yang sepenuhnya tergantung pada seluruh kolom yang membentuk kunci utama.

3. **Bentuk normal tahap ketiga (3^{rd} normal form)**

   Bentuk normal ketiga mengharuskan semua kolom pada tabel relasional tergantung hanya pada kunci utama. Secara definisi, sebuah tabel berada pada bentuk normal ketiga (3NF) jika tabel sudah berada

pada 2NF dan setiap kolom yang bukan kunci tidak tergantung secara transitif pada kunci utamanya.

4. ***Boyce Code Normal Form* (BCNF)**

   Setelah 3NF, semua masalah normalisasi hanya melibatkan tabel yang mempunyai tiga kolom atau lebih dan semua kolom adalah kunci. Banyak praktisi berpendapat bahwa menempatkan entitas pada 3NF sudah cukup karena sangat jarang entitas yang berada pada 3NF bukan merupakan 4NF dan 5NF.

5. **Bentuk Normal Tahap Keempat dan Kelima**

   Sebuah tabel relasional berada pada bentuk normal keempat (4NF) jika dia dalam BCNF dan semua ketergantungan multivalue merupakan ketergantungan fungsional. Bentuk normal keempat (4NF) didasarkan pada konsep ketergantungan multivalue (MVD).

   Sebuah tabel berada pada bentuk normal kelima (5NF) jika ia tidak dapat mempunyai dekomposisi lossless menjadi sejumlah tabel lebih kecil. Empat bentuk normal pertama berdasarkan pada konsep ketergantungan fungsional, sedangkan bentuk normal kelima berdasarkan pada konsep ketergantungan gabungan (*join dependence*) (Janner Simarmata ; 2010 : 76).

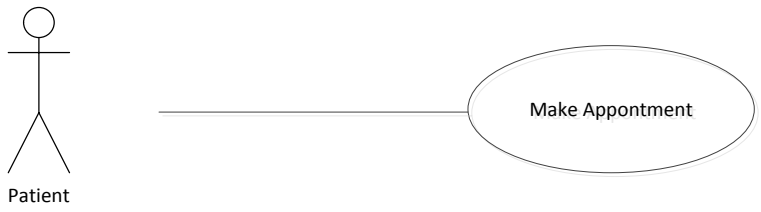
II.8. UML (*Unified Modeling Language*)

Menurut Arif Rachman (2012) Pemecahan masalah utama dari Object Oriented biasanya dengan penggambaran dalam bentuk model. Model abstrak (semu) merupakan gambaran detail dari inti masalah yang ada, umumnya sama seperti refleksi dari problem yang ada pada kenyataan. Beberapa modeling tool yang dipakai adalah bagian dari dasar UML, kependekan dari *United Modeling Language*. UML terdiri atas beberapa diagram, yaitu :

1. Diagram Use Case

Diagram Use Case menggambarkan apa saja aktifitas yang dilakukan oleh suatu sistem dari sudut pandang pengamatan luar. yang menjadi persoalan itu *apa yang dilakukan* bukan *bagaimana melakukannya*. Diagram Use Case dekat kaitannya dengan kejadian-kejadian. Kejadian (scenario) merupakan contoh apa yang terjadi ketika seseorang berinteraksi dengan sistem. untuk lebih memperjelas lihat gambaran suatu peristiwa untuk sebuah klinik kesehatan di bawah ini :

“Pasien menghubungi klinik untuk membuat janji (appointment) dalam pemeriksaan tahunan. Receptionist mendapatkan waktu yang luang pada buku jadwal dan memasukkan janji tersebut ke dalam waktu luang itu.”



Gambar II.2 Use Case Diagram
Sumber : (Arif Rachman ; 2012 : 2)

2. Diagram Class

Diagram Class memberikan pandangan secara luas dari suatu sistem dengan menunjukan kelas-kelasnya dan hubungan mereka. Diagram Class bersifat statis; *menggambarkan hubungan apa yang terjadi bukan apa yang terjadi jika mereka berhubungan*.

Multiplicity dari suatu titik *association* adalah angka kemungkinan bagian dari hubungan kelas dengan single *instance* (bagian) pada titik yang lain. *Multiplicity* berupa single number (angka tunggal) atau range number (angka batasan). Pada contoh, hanya bisa satu 'Customer' untuk setiap 'Order', tapi satu 'Customer' hanya bisamemiliki beberapa 'Order'. Tabel di bawah mengenai *multiplicity* yang sering digunakan :

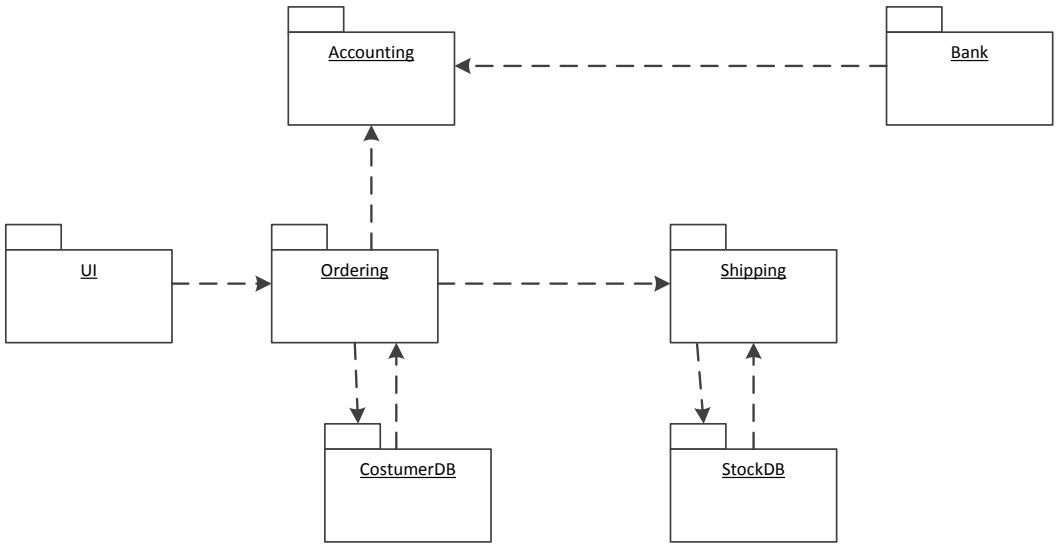
Tabel II.1 *Multiplicity*.

Multiplicity	Penjelasan
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimum 4

Sumber : (Arif Rachman ; 2012 : 3)

3. Package dan Object

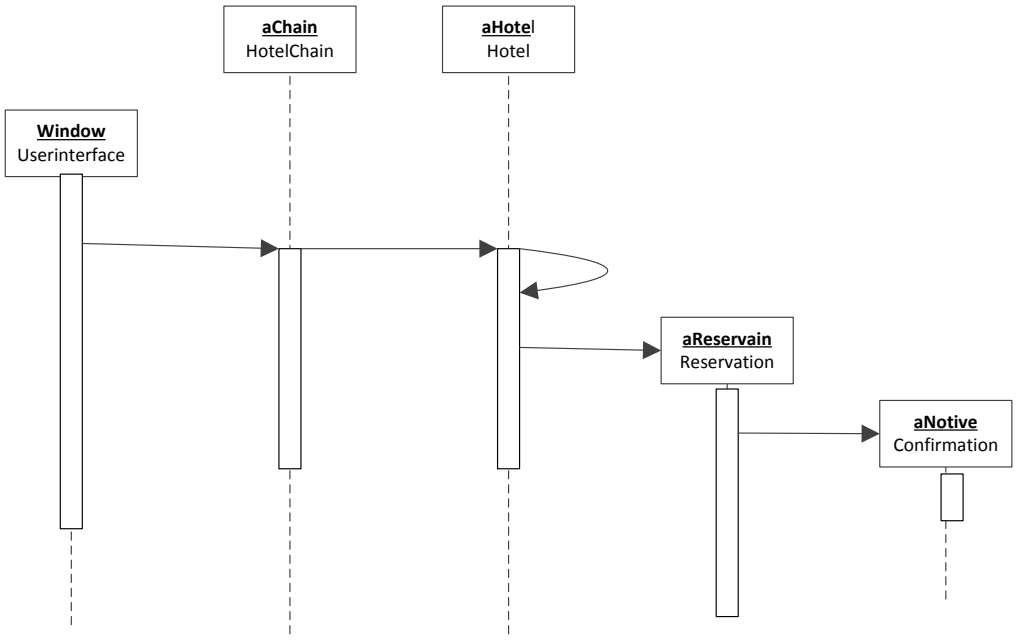
Untuk mengatur pengorganisasian diagram Class yang *kompleks*, dapat dilakukan pengelompokan kelas-kelas berupa *packag*e (paket-paket). *Package* adalah kumpulan elemen-elemen logika UML. Gambar di bawah ini mengenai model bisnis dengan pengelompokan kelas-kelas dalam bentuk paket-paket :



Gambar II.3. Diagram Package
Sumber : (Arif Rachman ; 2012 : 4)

4. Diagram Sequence

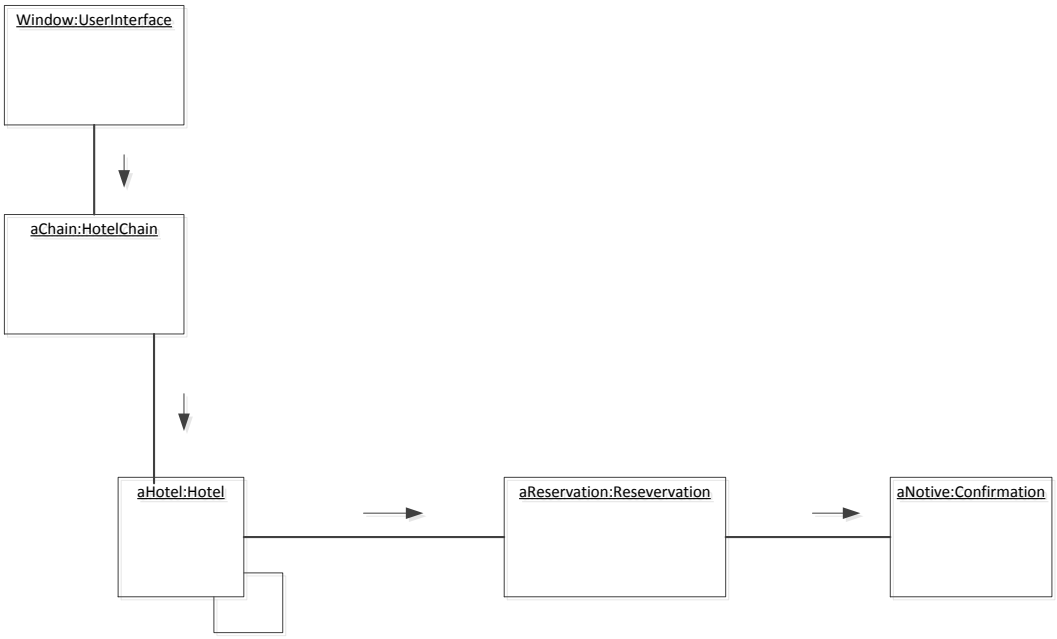
Diagram Class dan diagram Object merupakan suatu gambaran *model statis*. Namun ada juga yang bersifat *dinamis,* seperti Diagram Interaction. Diagram sequence merupakan salah satu diagram Interaction yang menjelaskan bagaimana suatu operasi itu dilakukan; *message* (pesan) apa yang dikirim dan kapan pelaksanaannya. Diagram ini diatur berdasarkan waktu. Obyek-obyek yang berkaitan dengan proses berjalannya operasi diurutkan dari kiri ke kanan berdasarkan waktu terjadinya dalam pesan yang terurut.



Gambar II.4. Diagram Sequence
Sumber : (Arif Rachman ; 2012 : 6)

5. Diagram Collaboration

Diagram Collaboration juga merupakan *diagram interaction*. Diagram membawa informasi yang sama dengan diagram Sequence, tetapi lebih memusatkan atau memfokuskan pada kegiatan obyek dari waktu pesan itu dikirimkan.

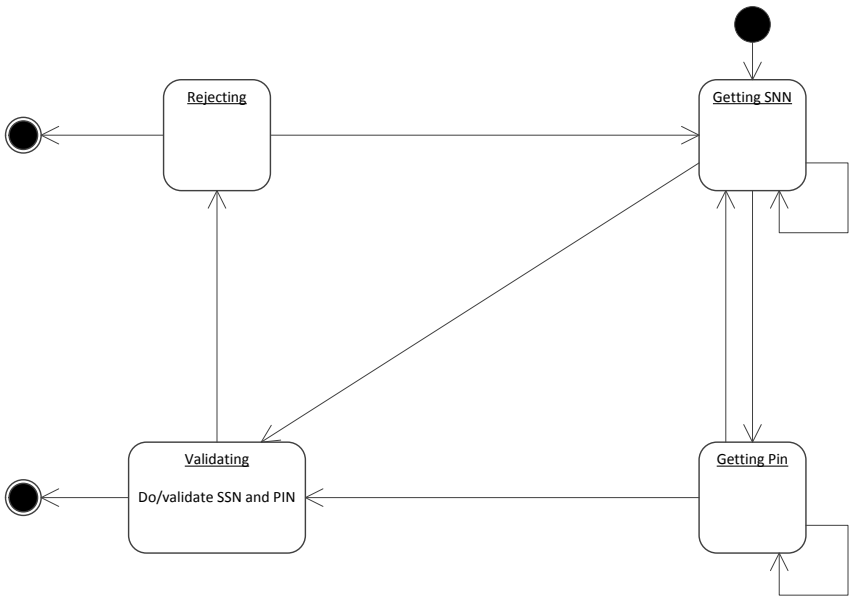


Gambar II.5. Diagram Collaboration
Sumber : (Arif Rachman ; 2012 : 7)

6. Diagram StateChart

Behaviors dan *state* dimiliki oleh obyek. Keadaan dari suatu obyek bergantung pada kegiatan dan keadaan yang berlaku pada saat itu. Diagram StateChart menunjukan kemungkinan dari keadaan obyek dan proses yang menyebabkan perubahan pada keadaannya.

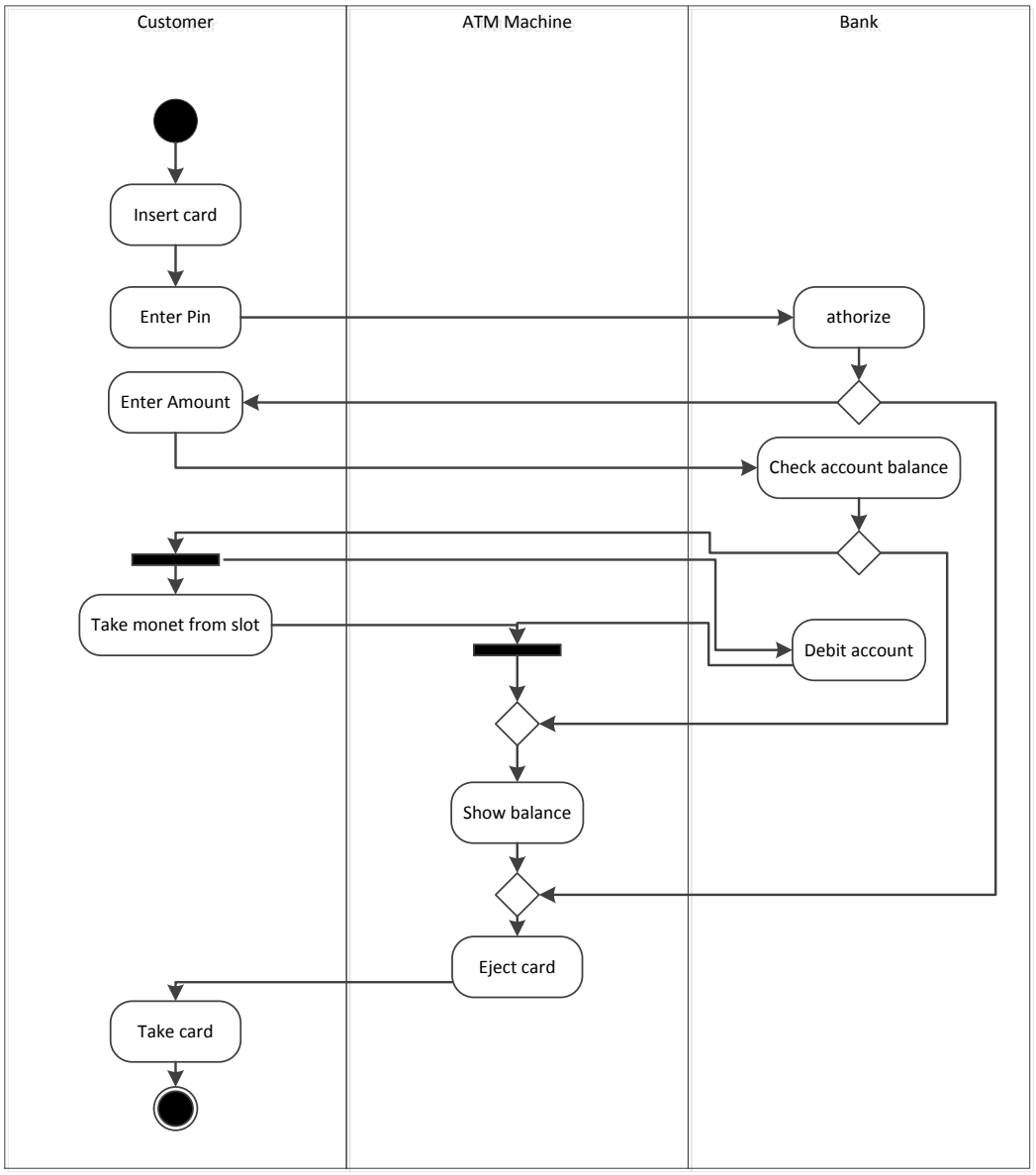
Untuk lebih jelas, contoh yang digunakan model diagram untuk login yang merupakan bagian dari Online Banking System. Logging in terdiri atas masukan input Social Security Number dan Personal Id Number yang berlaku, lalu memutuskan kesahan dari informasi tersebut.



Gambar II.6. Diagram Startchart
Sumber : (Arif Rachman ; 2012 : 8)

7. Diagram Activity

Pada dasarnya diagram Activity sering digunakan oleh *flowchart*. Diagram ini berhubungan dengan diagram Statechart. Diagram Statechart berfokus pada *obyek yang dalam suatu proses* (atau proses menjadi suatu obyek), diagram Activity berfokus pada *aktifitas-aktifitas yang terjadi yang terkait dalam suatu proses tunggal*. Jadi dengan kata lain, diagram ini menunjukkan bagaimana aktifitas-aktifitas tersebut bergantung satu sama lain.

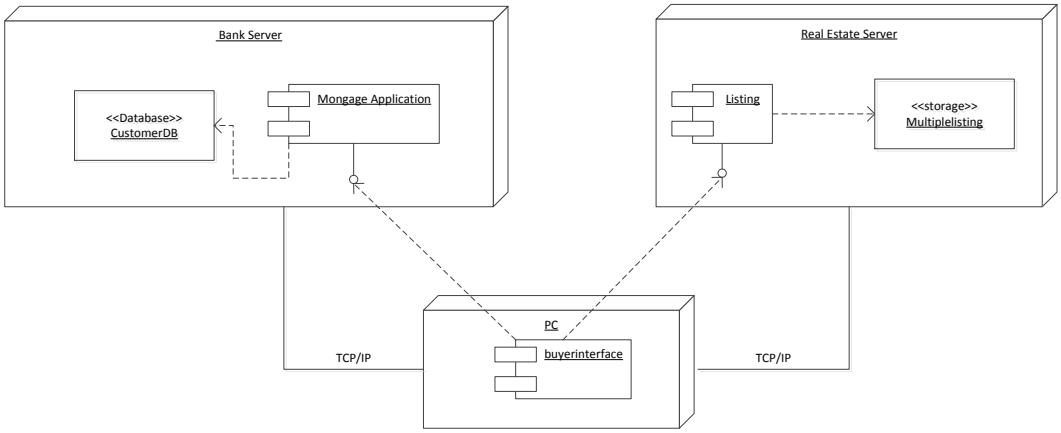


Gambar II.7. Diagram Activity
Sumber : (Arif Rachman ; 2012 : 9)

8. Diagram Component dan Deployment

Component adalah sebuah *code module* (kode-kode module). Diagram Component merupakan *fisik sebenarnya dari diagram Class*. Diagram Deployment menerangkan bahwa *konfigurasi fisik software* dan *hardware*. Gambar 2.10 menerangkan hubungan sekitar komponen software dan

hardware yang berperan dalam ruang lingkup real estate.



Gambar II.8. Diagram Component
Sumber : (Arif Rachman ; 2012 : 10)