BAB II

TINJAUAN PUSTAKA

2.1 Penelitian Rujukan

Telah banyak penelitian yang menerapkan SAW dalam berbagai objek permasalahan. Dibawah ini beberapa penelitian yang relevan dengan penelitian yang akan dilakukan :

1. Fajar Nugraha, Bayu Surarso dan Beta Noranita (2012), melakukan penelitian dengan judul ”*Sistem Pendukung Keputusan Evaluasi Pemilihan Pemenang Pengadaan Aset dengan Metode Simple Additive Weighting (SAW)*”. Kesimpulan dari penelitian ini bahwa metode *Simple Additive Weighting* (SAW) digunakan untuk mendukung pengambilan keputusan dalam proses evaluasi alternatif pemilihan pemenang pengadaan aset terutama dalam proses perangkingan berdasarkan kriteria-kriteria telah ditentukan sehingga dapat memberikan rekomendasi evaluasi pemilihan pemenang pengadaan aset yang lebih objektif karena dapat dilakukan pembobotan terhadap kriteria yang telah ditentukan.
2. Erick Ramdhani Sukaman (2014), melakukan penelitian dengan judul ”*Pendekatan Simple Additive Weighting (SAW) Untuk Menentukan Penerimaan Beasiswa Pada STIKOM Binaniaga Bogor*”. Kesimpulan dari penelitian ini bahwa *Simple Additive Weighting* (SAW) dapat direkomendasikan sebagai alat penunjang keputusan dalam menentukan penerima beasiswa.

3. Dyah Pratiwi, Juliana Putri Lestari dan Dewi Agushinta R (2014), melakukan penelitian dengan judul "*Decision Support System to Majoring High School Student Using Simple Additive Weighting Method*". Dengan kesimpulan bahwa "*Sistem yang telah dirancang dengan menerapkan SAW dapat membantu siswa dalam memilih jurusan. Implementasi kedalam bahasa pemrograman sangat mungkin dengan menerapkan berbagai kriteria yang ada*".

Sehubungan dengan penelitian sebelumnya tentang metode yang digunakan, maka akan dilakukan penelitian tentang pendekatan SAW untuk seleksi penerima beasiswa. Hasil yang diharapkan yaitu menentukan penerima produk Gold yang tepat dan sesuai.

2.2 Gadai

Menurut kitab Undang- Undang Hukum Perdata pasal 1150 disebutkan bahwa gadai adalah suatu hak yang diperoleh seorang yang berpiutang atas suatu barang bergerak, dan yang memberikan kekuasaan kepada orang berpiutang itu untuk mengambil pelunasan dari barang tersebut secara didahulukan dari pada orang yang berpiutang lainya; dengan pengecualian biaya untuk melelang barang tersebut dan biaya yang telah dikeluarkan untuk menyelamatkan barang itu setelah digadaikan, biaya- biaya mana yang harus didahulukan.

Secara umum usaha gadai adalah kegiatan menjaminkan barang- barang berharga kepada kepada pihak tertentu, guna memperoleh sejumlah uang

dan barang yang dijaminkan akan ditebus kembali sesuai perjanjian antara nasabah dengan lembaga gadai. Pegadaian terdiri dari dua macam, yaitu pegadaian konvensional dan pegadaian syariah. Pegadaian adalah lembaga yang melakukan pembiayaan dengan bentuk penyaluran kredit atas dasar hukum kredit. Dengan demikian, dari pengertian diatas dapat disimpulkan bahwa usaha gadai memiliki ciri- ciri diantaranya:

1. Terdapat barang – barang berharga yang digadaikan
2. Nilai jumlah pinjaman tergantung nilai barang yang digadaikan
3. Barang yang digadaikan dapat ditebus kembali

2.3 Produk Gold

Produk Gold adalah suatu penawaran yang diberikan untuk mengapresiasi nasabah dari PT. Pinjam Indonesia. Produk ini diberikan berdasarkan catatan/*track record* dari suatu anggota selama enam bulan. Dalam enam bulan tersebut hanya dua orang yang berhak mendapatkan produk Gold yaitu potongan 10% dari total tagihan.

2.4 XAMPP

Server HTTP Apache atau Server Web Apache adalah server web yang dapat dijalankan di banyak sistem operasi (Unix, BSD, Linux, Windows, dan Novell Netware serta platform lainya) yang berupa untuk melayani dan memfungsikan situs web. Protokol yang digunakan untuk melayani fasilitas web ini menggunkan HTTP.

2.4.1 Apache

Apache memiliki fitur-fitur seperti pesan kesalahan yang dapat dikonfigurasi, otentifikasi berbasis basis data dan lain-lain. Apache juga didukung oleh sejumlah antarmuka pengguna berbasis grafik (GUI) yang memungkinkan penanganan server menjadi mudah.

Apache adalah komponen server web dari paket perangkat lunak LAMP (Linux, Apache, MySQL, PHP/Perl/Python). Karena berbagai keunggulan dan kelebihan yang dimiliki web server apache, server web ini menjadi sebuah web server yang paling populer dikalangan pengguna dengan kelebihan sebagai berikut :

1. Open Source, Free software
2. Apache dapat berjalan di beberapa sistem operasi (Unix, BSD, Linux, Microsoft, Novell).
3. Apache memiliki fitur-fitur canggih seperti pesan kesalhan yang dapat dikonfigurasi, autentifikasi berbasis basis data dan lain-lain. Apache jugag didukung oleh sejumlah antarmuka pengguna berbasis grafik (GUI).
4. Fleksibel , mudah settinganya (Fleksibilitas untuk di setting dengan PHP dan MySQL).
5. Kehandalannya telah teruji.

2.4.2 MySQL

PHP MyAdmin adalah sebuah aplikasi open source yang berfungsi untuk memudahkan manajemen MySQL. Dengan mengunakan

phpmyadmin, anda dapat membuat database, membuat tabel, menginsert, menghapus dan mengupdate data dengan GUI dan terasa lebih mudah, tanpa perlu mengetikkan perintah SQL secara manual. PHPMyAdmin dapat di download secara free di http://www.phpmyadmin.net. Karena berbasis web, maka phpmyadmin dapat di jalankan di banyak sistem operasi, selama dapat menjalankan webserver dan Mysql.

2.5 Hypertext Preprocessor (PHP)

2.5.1 Pengertian PHP

PHP pertama kali dibuat oleh Rasmus Lerdorf pada tahun 1995. Pada waktu itu PHP bernama FI (*Form Interpreted*). Pada saat tersebut PHP adalah sekumpulan script yang digunakan untuk mengolah data form dari web.

Berdasarkan pendapat para ahli yang dapat ditarik kesimpulan bahwa PHP adalah bahasa pemrograman yang digunakan secara luas untuk penanganan pembuatan dan pengembangan sebuah situs web dan bisa digunakan bersamaan dengan HTML.

PHP pada awalnya diperkenalkan sebagai singkatan dari Hypertext Preprocessor. PHP pertama ditulis menggunakan bahasa Perl (Perl Script), kemudian ditulis ulang menggunakan bahasa pemograman C CGI-BIN (Common Gateway Interface-Binary) yang ditujukan untuk mengembangkan halaman website yang mendukung formulir dan penyimpanan data. Pada tahun 1995 PHP Tool 1.0 dirilis untuk umum, kemudian pengembangannya dilanjutkan oleh Andi Gutmans dan Zeev

Suraski. Perusahaan bernama Zend kemudian melanjutkan pengembangan PHP dan merilis PHP versi 5 terakhir pada terakhir pada saat ini. Aplikasi bahasa PHP dapat dipergunakan sebagai landasan operasi pada pemrogaman jaringan berbasis web, digunakan juga untuk pemrogaman database, dan digunakan untuk membuat aplikasi web.

2.5.2 HTML (Hyper Text Markup Language)

HTML *(Hyper Text Markup Language)* adalah sebuah bahasa *markup* yang digunakan untuk membuat sebuah halaman web dan menampilkan berbagai informasi di dalam sebuah *browser* internet. Bermula dari sebuah bahasa yang sebelumnya banyak digunakan di dunia penerbitan dan percetakan yang disebut dengan SGML *(Standard Generalized Markup Language),* HTML adalah sebuah standar yang digunakan secara luas untuk menampilkan halaman web. HTML saat ini dikendalikan penggunaannya oleh *World Wide Web Consortium* (W3C).

2.6 Code Igniter

CodeIgniter adalah aplikasi open source yang berupa framework dengan model MVC (Model, View, Controller) untuk membangun website dinamis dengan menggunakan PHP. CodeIgniter memudahkan developer untuk membuat aplikasi web dengan cepat dan mudah dibandingkan dengan membuatnya dari awal. Codeigniter adalah sebuah aplikasi open source yang bebas untuk digunakan oleh siapapun tanpa harus membayar lisensi untuk menggunakannya.

Framework secara sederhana dapat diartikan kumpulan dari fungsi /prosedur dan *class* untuk tujuan tertentu yang sudah siap digunakan sehingga bisa lebih mempermudah dan mempercepat pekerjaan seorang pemrograman, tanpa harus membuat fungsi atau class dari awal.

Ada beberapa alasan mengapa menggunakan Framework:

1. Mempercepat dan mempermudah untuk membangun sebuah website atau aplikasi web.
2. Proses maintenance lebih mudah karena sudah ada skema tertentu dalam sebuah framework.
3. Secara umum framework menyediakan fasilitas-fasilitas yang umum dipakai sehingga kita tidak perlu membangun dari awal (misalnya validasi, *pagination*, *multiple database*, *scaffolding*, *session*, *error handling*, dsb).

2.7 *Simple Additive Weighting* (SAW)

Metode *Simple Additive Weighting* (SAW) sering juga dikenal istilah metode penjumlahan terbobot. Konsep dasar metode SAW adalah mencari penjumlahan terbobot dari rating kinerja pada setiap alternatif pada semua atribut (Fishburn, 1967) (MacCrimmon, 1968). Metode SAW membutuhkan proses normalisasi matriks keputusan (X) ke suatu skala yang dapat diperbandingkan dengan semua rating alternatif yang ada.

$$r_{ij} = \begin{cases} \dfrac{x_{ij}}{Max_{ij}(x_{ij})} & Jika\ j\ adalah\ kriteria\ keuntungan\ (benefit) \\ \dfrac{Min_{ij}(x_{ij})}{x_{ij}} & Jika\ j\ adalah\ kriteria\ biaya\ (cost) \end{cases}$$

Keterangan :

1. Dikatakan kriteria keuntungan apabila nilai x_{ij} memberikan keuntungan bagi pengambil keputusan, sebaliknya kriteria biaya apabila x_{ij} menimbulkan biaya bagi pengambil keputusan
2. Apabila berupa kriteria keuntungan maka nilai x_{ij} dibagi dengan nilai $Max_i(x_{ij})$ dari setiap kolom, sedangkan untuk kriteria biaya, nilai $Min_i(x_{ij})$ dari setiap kolom dibagi dengan nilai x_{ij}

Nilai Preferensi untuk setiap alternative (Vi) diberikan sebagai :

$$Vi = \sum_{j=1}^{n} w_j r_{ij}$$

Keterangan :

Vi = ranking untuk setiap alternative

w_j = nilai bobot dari setiap kriteria

r_{ij} = nilai rating kinerja ternormalisasi

Nilai Vi yang lebih besar mengindikasikan bahwa alternative A, merupakan alternatif terbaik.

Langkah Perhitungan Metode SAW

1. Menentukan alternatif, yaitu A_i.
2. Menentukan kriteria yang akan dijadikan acuan dalam pengambilan keputusan yaitu C_j.
3. Menentukan bobot preferensi atau tingkat kepentingan (*w*) setiap kriteria.

$w = [\ w_1\ w_2\ w_3\ \ldots.\ w_j\ ]$

4. Membuat tabel rating kecocokan setiap alternatif pada setiap kriteria.
5. Membuat matrik keputusan X yang dibentuk dari tabel rating kecocokan dari setiap alternatif pada setiap kriteria. Nilai {x} setiap alternatif (Ai) pada setiap kriteria (Cj) yang sudah ditentukan dimana, I = 1,2,…,m dan j = 1,2,…,n.
6. Melakukan normalisasi matrik keputusan X dengan cara menghitung nilai rating kinerja ternormalisasi (r_{ij}) dari alternative (A_i) pada kriteria (C_j).
7. Hasil dari nilai rating kinerja ternormalisasi (r_{ij}) membentuk matrik ternormalisasi (R).

$$R = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1j} \\ \vdots & & & \vdots \\ r_{i1} & r_{i2} & \cdots & r_{ij} \end{bmatrix}$$

8. Hasil akhir nilai preferensi (V_i) diperoleh dari penjumlahan dari perkalian elemen baris matrik ternormalisasi (r) dengan bobot preferensi (w) yang bersesuaian elemen kolom matrik (w).

Berikut contoh penerapan SAW :

Suatu perusahaan di Jakarta ingin membangun sebuah gudang yang akan digunakan sebagai tempat untuk menyimpan sementara hasil produksinya. Ada 3 lokasi yang akan menjadi alternatif, yaitu A1 = Tanjung Priuk, A2 = Pulo Gadung, A3 = Kali Deres. Ada 5 kriteria yang dijadikan acuan dalam pengambilan keputusan yaitu :

1. C1 = Jarak dengan pasar terdekat (km)
2. C2 = Kepadatan penduduk di sekitar lokasi (orang/km^2)
3. C3 = Jarak dari pabrik (km)
4. C4 = jarak dengan gudang yang sudah ada (km)
5. C5 = Harga tanah untuk lokasi

Rating kecocokan setiap alternatif pada setiap kriteria, dinilai dengan 1 sampai 5, yaitu 1 = sangat buruk, 2 = buruk, 3 = cukup, 4 = baik, 5 = sangat baik. Pengambilan keputusan memberikan bobot tiap kriteria sebagai beriku : W = (5,3,4,4,2).

Tabel 2.1 Tabel Rating Kecocokan

Alternatif	Kriteria				
	C1	C2	C3	C4	C5
A1	4	4	5	3	3
A2	3	3	4	2	3
A3	5	4	2	2	2

Kemudian dari data pada Tabel 2.1 dibentuk sebuah matriks dari setiap alternatif pada setiap kriteria

$$X = \begin{bmatrix} 4 & 4 & 5 & 3 & 3 \\ 3 & 3 & 4 & 2 & 3 \\ 5 & 4 & 2 & 2 & 2 \end{bmatrix}$$

Kemudian dilakukan proses normalisasi matrik keputusan dengan cara menghitung rating kinerja ternormalisasi (r_{ij}) berdasarkan persamaan yang disesuaikan dengan jenis kriteria.

$$r_{11} = \frac{4}{Max(4,3,5)} = \frac{4}{5} = 0.8$$

$$r_{21} = \frac{3}{Max(4,3,5)} = \frac{3}{5} = 0.6$$

$$r_{31} = \frac{5}{Max(4,3,5)} = \frac{5}{5} = 1$$

$$r_{12} = \frac{4}{Max(4,3,4)} = \frac{4}{4} = 1$$

$$r_{22} = \frac{3}{Max(4,3,4)} = \frac{3}{4} = 0.75$$

$$r_{32} = \frac{4}{Max(4,3,4)} = \frac{4}{4} = 1$$

$$r_{13} = \frac{5}{Max(5,4,2)} = \frac{5}{5} = 1$$

$$r_{23} = \frac{4}{Max(5,4,2)} = \frac{4}{5} = 0.8$$

$$r_{33} = \frac{2}{Max(5,4,2)} = \frac{2}{5} = 0.4$$

$$r_{14} = \frac{3}{Max(3,2,2)} = \frac{3}{3} = 1$$

$$r_{24} = \frac{2}{Max(3,2,2)} = \frac{2}{3} = 0.7$$

$$r_{34} = \frac{2}{Max(3,2,2)} = \frac{2}{3} = 0.7$$

$$r_{15} = \frac{3}{Max(3,3,2)} = \frac{3}{3} = 1$$

$$r_{25} = \frac{3}{Max(3,3,2)} = \frac{3}{3} = 1$$

$$r_{35} = \frac{2}{Max(3,3,2)} = \frac{2}{3} = 0.7$$

Hasil dari rating kinerja ternormalisasi akan membentuk matriks ternormalisasi seperti di bawah ini

$$X = \begin{bmatrix} 0.8 & 1 & 1 & 1 & 1 \\ 0.6 & 0.75 & 0.8 & 0.7 & 1 \\ 1 & 1 & 0.4 & 0.7 & 0.7 \end{bmatrix}$$

Setelah mendapatkan matriks ternormalisasi akan dilakukan perangkingan dengan mengkalikan bobot yang ditetapkan diawal.

V_1 = {(5)(0.8) + (3)(1) + (4)(1) + (4)(1) + (2)(1)} = 17

V_2 = {(5)(0.6) + (3)(0.75) + (4)(0.8) + (4)(0.7) + (2)(1)} = 13.25

V_3 = {(5)(1) + (3)(1) + (4)(0.4) + (4)(0.7) + (2)(0.7)} = 13.8

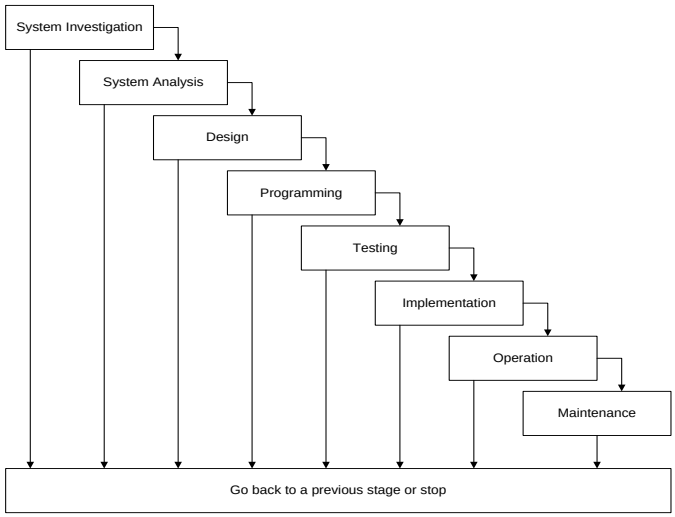
Jadi nilai terbesar adalah V_1 sehingga A_1 adalah alternatif yang dipilih sebagai alternatif terbaik.

2.8 Model Pengembangan Perangkat Lunak

Pada rekayasa perangkat lunak, banyak model yang telah dikembangkan untuk membantu proses pengembangan perangkat lunak. Model-model ini pada umumnya mengacu pada model proses pengembangan sistem yang disebut *System Development Life Cycle* (SDLC).

SDLC adalah tahapan-tahapan pekerjaan yang dilakukan oleh analis sistem dan *programmer* dalam membangun sistem informasi. Jumlah langkah SDLC pada referensi lain mungkin berbeda, namun secara umum adalah sama, antara lain:

Contoh penerapan SDLC adalah model air terjun atau disebut juga dengan model *Waterfall*. Tahap-tahapnya adalah sebagai berikut:



Gambar 2.1 System Development Life Cycle (SDLC)

2.8.1 System Investigation

Pada tahap ini, dilakukan studi kelayakan untuk menentukan kemungkinan suksesnya proyek pengembangan sistem yang diajukan dan menaksir kelayakan proyek dari segi teknis, ekonomis dan perilaku.

2.8.2 System Analysis

Software merupakan bagian dari sebuah sistem yang besar, maka pengerjaan dimulai dengan mengumpulkan kebutuhan bagi semua elemen-elemen sistem kemudian mengalokasikan beberapa subset dari kebutuhan-kebutuhan tersebut dalam *software*. Hal ini sangat penting ketika *software* harus berhubungan dengan elemen lain seperti *hardware*, manusia dan basis data. Tahap ini meliputi pengumpulan kebutuhan pada tingkat sistem dengan sedikit analisa dan perancangan ditingkat atas. Proses pengumpulan elemen sistem ditingkatkan dan dipusatkan secara khusus

pada *software* untuk mengerti karakteristik dari program yang akan dibuat. Sistem analisis *software* harus mengerti ruang lingkup informasi yang ingin dicakup dalam pembuatan *software* seperti fungsi-fungsi yang dibutuhkan, karakteristik, kinerja dan tampilan.

2.8.3 Design

Adalah proses bertahap yang berfokus pada 4 atribut program yang berbeda yaitu struktur database, arsitektur perangkat lunak, implementasi kebutuhan dan detail *procedural* (algoritma). Proses perancangan menerjemahkan kebutuhan elemen sistem yang direpresentasikan ke dalam suatu *software* yang diperkirakan kualitasnya sebelum dilakukan pengkodean.

2.8.4 Programming

Pada tahap ini rancangan diterjemahkan ke dalam bentuk yang dapat dibaca oleh mesin jika perancangan dilaksanakan secara detail. Pengkodean dapat dilakukan secara mekanis.

2.8.5 Testing

Jika kode telah dibuat atau ditulis maka diadakan pengujian program. Pengujian juga dilakukan untuk memastikan masukan yang diberikan menghasilkan keluaran yang diinginkan.

2.8.6 Implementation and Operation

Setelah implementasi, sistem baru akan beroperasi dalam periode tertentu. Saat operasi sistem baru telah stabil, audit dilakukan guna menaksir kapabilitas sistem dan menentukan apakah penggunaanya tepat.

2.8.7 Maintenance

Software yang telah diimplementasikan dapat mengalami perubahan yang disebabkan oleh penemuan *bug* ataupun penyesuaian akibat perubahan di lingkungan implementasi (perubahan hardware ataupun kebutuhan pengguna). Dengan adanya pemeliharaan *software*, perubahan yang terjadi akan lebih mudah dilakukan dibandingkan harus membuat ulang program baru.

2.9 *Unified Modeling Language* (UML)

UML (*Unified Modelling Language*) adalah sebuah bahasa yang menjadi standar dalam industri untuk memvisualisasikan, merancang dan mendokumentasikan *system* piranti lunak. Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi *software*, dimana aplikasi tersebut dapat berjalan pada *hardware*, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun.

Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax* semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan.

2.9.1 Konsep pemodelan menggunakan UML

Pemodelan menggunakan *Unified Modeling Language* (UML) merupakan metode pemodelan berorientasi objek dan berbasis visual. Karenanya pemodelan menggunakan UML merupakan pemodelan objek

yang fokus pada pendefinisian struktur statis dan model sistem informasi yang dinamis daripada mendefinisikan data dan model proses yang tujuannya adalah pengembangan tradisional. Terdapat berbagai diagram dalam merancang *system* menggunakan UML, diantaranya adalah :

1. **Use Case Diagram**

   Use Case Diagram secara grafis menggambarkan interaksi antara sistem, sistem eksternal, dan pengguna. Dengan kata lain *Use Case Diagram* secara grafis mendeskripsikan siapa yang akan menggunakan sistem dan dalam cara apa pengguna (*user*) mengharapkan interaksi dengan sistem itu.

2. **Class Diagram**

   Class Diagram menggambarkan struktur *object* sistem. Diagram ini menunjukan *class object* yang menyusun sistem dan juga hubungan antara *class object* tersebut.

3. ***Sequence Diagram***

   Sequence Diagram secara grafis menggambarkan bagaimana objek berinteraksi dengan satu sama lain melalui pesan pada sekuensi sebuah *use case* atau operasi. Diagram ini mengilustrasikan bagaimana pesan terkirim dan diterima di antara objek dan dalam sekuensi atau *timing* apa.

4. **Diagram kolaborasi/ *Collaboration Diagram***

   Diagram kolaborasi/ *Collaboration Diagram* serupa dengan diagram rangkaian/ sekuensi, tetapi tidak fokus pada *timing* atau sekuensi

pesan. Diagram ini justru menggambarkan interaksi (atau kolaborasi) antara objek dalam sebuah format jaringan.

5. *Statechart Diagram*

Diagram statechart digunakan untuk memodelkan *behavior* objek khusus yang dinamis. Diagram ini mengilustrasikan siklus hidup objek berbagai keadaan yang dapat diasumsikan oleh objek dan *event* (kejadian) yang menyebabkan objek beralih dari satu *state* ke *state* lain.

6. *Activity Diagram*

Activity Diagram secara grafis digunakan untuk menggambarkan rangkaian aliran aktivitas baik proses bisnis maupun *use case*. *Activity diagram* dapat juga digunakan untuk memodelkan *action* yang akan dilakukan saat sebuah operasi di eksekusi, dan memodelkan hasil dari *action* tersebut.

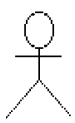
7. Collaboration Diagram

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.

2.9.2 Notasi UML

Notasi UML merupakan sekumpulan bentuk khusus untuk mengambarakan berbagai diagram piranti lunak.

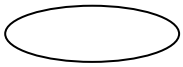
1. *Actor*



Gambar 2.2 Notasi Actor

Actor menggambarkan segala pengguna *software* aplikasi (*user*). *Actor* memberikan suatu gambaran jelas tentang apa yang harus dikerjakan *software* aplikasi. Sebagai contoh sebuah *actor* dapat memberikan *input* ke dalam dan menerima informasi dari *software* aplikasi, perlu dicatat bahwa sebuah *actor* berinteraksi dengan *use case*, tetapi tidak memiliki kontrol atas *use case*. Sebuah *actor* mungkin seorang manusia, satu *device, hardware* atau sistem informasi lainnya.

2. *Use Case*



Gambar 2.3 Notasi *Use Case*

Use case menjelaskan urutan kegiatan yang dilakukan *actor* dan sistem untuk mencapai suatu tujuan tertentu. Walaupun menjelaskan kegiatan, namun *use case* hanya menjelaskan apa yang dilakukan oleh *actor* dan sistem bukan bagaimana *actor* dan sistem melakukan kegiatan tersebut.

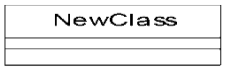
A. *Use-case* Konkret adalah *use case* yang dibuat langsung karena keperluan *actor*. *Actor* dapat melihat dan berinisiatif terhadapnya.
B. *Use-case* Abstrak adalah *use case* yang tidak pernah berdiri sendiri. *Use case abstrak* senantiasa termasuk di dalam (*include*), diperluas dari (*extend*) atau memperumum (*generalize*) *use case* lainnya.

Untuk menggambarkannya dalam *use case* model biasanya digunakan *association relationship* yang memiliki *stereotype include, extend* atau *generalization relationship*. Hubungan *include* menggambarkan bahwa suatu *use case* seluruhnya meliputi fungsionalitas

dari *use case* lainnya. Hubungan *extend* antar *use case* berarti bahwa satu *use case* merupakan tambahan fungsionalitas dari *use case* yang lain jika kondisi atau syarat tertentu terpenuhi.

3. *Class*

Class merupakan pembentuk utama dari sistem berorientasi objek, karena *class* menunjukan kumpulan objek yang memiliki atribut dan operasi yang sama. *Class* digunakan untuk mengimplementasikan *interface*.

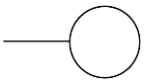


Gambar 2.4 Notasi *Class*

Class digunakan untuk mengabstraksikan elemen-elemen dari sistem yang sedang dibangun. *Class* bisa merepresentasikan baik perangkat lunak maupun perangkat keras, baik konsep maupun benda nyata. Notasi *class* berbentuk persegi panjang berisi 3 bagian persegi panjang paling atas untuk *namaclass*, persegi panjang paling bawah untuk *operasi,* dan persegi panjang ditengah untuk *atribut.*

Atribut digunakan untuk menyimpan informasi. Nama atribut menggunakan kata benda yang bisa dengan jelas merepresentasikan informasi yang tersimpan di dalamnya. Operasi menunjukan sesuatu yang bisa dilakukan oleh objek dan menggunakan kata kerja.

4. *Interface*



Gambar 2.5 Notasi *Interface*

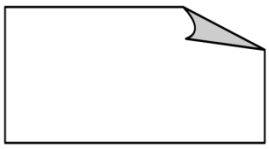
Interface merupakan kumpulan operasi tanpa implementasi dari suatu *class*. Implementasi operasi dalam *interface* dijabarkan oleh operasi di dalam *class*. Oleh karena itu keberadaan *interface* selalu disertai oleh *class* yang mengimplementasikan operasinya. *Interface* ini merupakan salah satu cara mewujudkan *prinsip enkapsulasi* dalam objek.

5. *Interaction*

Gambar 2.6 Notasi *Interaction*

Interaction digunakan untuk menunjukan baik aliran pesan atau informasi antar objek maupun hubungan antar objek. Biasanya interaction ini dilengkapi juga dengan teks bernama *operation signature* yang tersusun dari nama operasi, parameter yang dikirim dan tipe parameter yang di kembalikan.

6. *Note*



Gambar 2.7 Notasi *Note*

Note digunakan untuk memberikan keterangan atau komentar tambahan dari suatu elemen sehingga bisa langsung terlampir dalam model. *Note* ini bisa disertakan ke semua elemen notasi yang lain.

7. *Dependency*

Gambar 2.8 Notasi *Dependency*

Dependency merupakan relasi yang menunjukan bahwa perubahan pada salah satu elemen memberi pengaruh pada elemen lain. Elemen yang ada di bagian tanda panah adalah elemen yang tergantung pada elemen yang ada di bagian tanpa tanda panah.

Terdapat 2 *stereotype* dari *dependency*, yaitu *include* dan *extend*. *Include* menunjukan bahwa suatu bagian dari elemen (yang ada di garis tanpa panah) memicu eksekusi bagian dari elemen lain (yang ada di garis dengan panah). *Extend* menunjukan bahwa suatu bagian dari elemen di garis tanpa panah bisa disisipkan ke dalam elemen yang ada di garis dengan panah.

8. *Association*

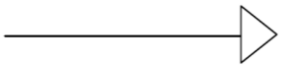
Gambar 2.9 Notasi *Association*

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

Navigation dilambangkan dengan penambahan tanda panah di akhir garis. *Bidirectional navigation* menunjukan bahwa dengan mengetahui salah satu *class* bisa di dapatkan informasi dari *class* lainnya. Sementara *UniDirectional navigation* hanya dengan mengetahui *class* di ujung garis *association* tanpa panah kita bisa mendapatkan informasi dari *class* di ujung dengan panah, tetapi tidak sebaliknya. *Aggregation* mengacu pada

hubungan “has-a”, yaitu bahwa suatu *class* memiliki *class* lain, misalnya Rumah memiliki *class* Kamar.

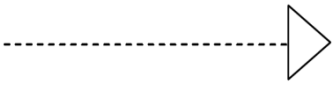
9. *Generalization*



Gambar 2.10 Notasi *Generalization*

Generalization menunjukan hubungan antara elemen yang lebih umum ke elemen yang lebih spesifik. Dengan *generalization*, *class* yang lebih spesifik (*subclass*) akan menurunkan atribut dan operasi dari *class* yang lebih umum (*superclass*) atau *“subclass is superclass”.* Dengan menggunakan notasi *generalization* ini, konsep *inheritance* dari prinsip hirarki dapat dimodelkan.

10. *Realization*



Gambar 2.11 Notasi *Realization*

Realization menunjukan hubungan bahwa elemen yang ada di bagian tanpa panah akan merealisasikan apa yang dinyatakan oleh elemen yang ada di bagian dengan panah. Misalnya *class* merealisasikan *package*, *component* merealisasikan *class* atau *interface*.