**BAB II**

**LANDASAN TEORI**

Teknologi dan manusia tidak akan pernah bisa dipisahkan. Manusia selamanya akan tetap membutuhkan teknologi dalam memudahkan aktivitasnya. Komputer adalah salah satu bukti kemajuan peradaban teknologi informasi yang memudahkan manusia dalam melakukan pekerjaannya. Seiring dengan berkembangnya teknologi, komputer yang dulunya memiliki ukuran sangat besar sehingga memenuhi ruangan kini telah diubah menjadi ukuran yang lebih kecil. Dengan memanfaatkan teknologi *microprocessor*, laptop atau komputer jinjing ini dapat mudah dibawa dan tidak membutuhkan tempat yang luas untuk penyimpanannya.

# 2.1 Laptop

Menurut Setianto dalam bukunya “Serba-Serbi Laptop” menjelaskan laptop atau sering disebut juga *notebook* adalah komputer bergerak yang berukuran relatif kecil dan ringan, beratnya berkisar dari 1 hingg 6 kg, tergantung ukuran, bahan, dan spesifikasi laptop tersebut. Sumber daya laptop berasal dari baterai atau adaptor A/C yang dapat digunakan untuk mengisi ulang baterai dan menyalakan laptop itu sendiri. Baterai laptop pada umumnya dapat bertahan sekitar 1 hingga 6 jam, tergantung dari cara pemakaian, spesifikasi, dan ukuran baterai. Komponen yang terdapat di dalamnya sama persis dengan komponen pada desktop, hanya saja ukurannya diperkecil, dijadikan lebih ringan, lebih tidak panas dan lebih hemat daya. Laptop kebanyakan menggunakan layar LCD (*Liquid Crystal Display*) berukuran 10 *inch* hingga 17 *inch* tergantung dari ukuran laptop itu sendiri.

Berbeda dengan komputer *desktop*, laptop memiliki komponen pendukung yang didesain secara khusus untuk mengakomodasi sifat laptop yang *portable*. Sifat utama yang dimiliki oleh komponen penyusun laptop adalah ukuran yang kecil, hemat konsumsi energi, dan efisien. (Suratma, 2012)

7

Berdasarkan hasil wawancara terhadap pemilik toko *Image Computer* maka didapatkan beberapa spesifikasi laptop yang jadi pertimbangan saat pembeli mau membeli laptop adalah sebagai berikut:

1. Harga

Pada dasarnya adalah tolak ukur terpenting bagi sebagian besar calon pembeli, khususnya yang berada pada tingkat ekonomi menengah kebawah.

# 2. RAM (*Random Access Memory*)

RAM sebagai media penyimpanan temporer dalam sebuah komputer, menjadi pertimbangan penting, dikarenakan RAM sangat mempengaruhi kinerja dari sebuah komputer. Semakin besar kapasitas dan semakin tinggi kecepatan dari RAM sebuah laptop, semakin bagus dan cepat pula kinerja laptop tersebut.

# *3. Processor*

*Processor* sebagai otak dari komputer merupakan salah satu pertimbangan penting dalam pemilihan laptop. *Processor* dengan kecepatan yang tinggi mampu memproses dan melakukan perhitungan dengan cepat pula.

# 4. *Harddisk*

*Harddisk* sebagai media penyimpanan semi-permanen menjadi pertimbangan penting, dimana semakin besar kapasitas *harddisk* sebuah laptop, semakin banyak pula data-data yang bisa disimpan oleh penggunanya.

5. VGA (*Video Graphics Array*).

VGA sebagai pengolah data grafis dalam sebuah laptop menjadi pertimbangan penting, khususnya bagi calon pembeli laptop yang bertujuan untuk menggunakan laptopnya sebagai media bermain *game* ataupun sebagai media bekerja yang menggunakan aplikasi-aplikasi multimedia yang berat, contohnya *Photoshop, Corel Draw, Auto* CAD , dan sejenisnya.

## 2.2 Sistem Penunjang Keputusan (SPK)

Pada dasarnya sistem pendukung keputusan merupakan pengembangan lebih lanjut dari sistem informasi manajemen terkomputerisasi yang dirancang sedemikian rupa sehingga bersifat interaktif dengan pemakainya. Sifat interaktif dimaksudkan untuk memudahkan integrasi antara berbagai komponen dalam proses pengambilan keputusan seperti prosedur, kebijakan, teknik analisis, serta pengalaman dan wawasan manajerial guna membentuk suatu kerangka keputusan

bersifat fleksibel.

SPK atau *Decision Support Sistem* (DSS) pertama kali diungkapkan pada awal tahun 1970-an oleh Michael S. Scott Morton dengan istilah *Management Decision Sistem*. Sistem tersebut adalah suatu sistem yang berbasis komputer yang ditujukan untuk membantu mengambil keputusan dengan memanfaatkan data dan model tertentu untuk memecahkan berbagai persoalan yang tidak terstruktur.

SPK adalah sebuah sistem yang mampu memberikan kemampuan pemecahan masalah maupun kemampuan pengkomunikasian untuk masalah

dengan kondisi semi terstruktur dan tak terstruktur. Sistem ini digunakan untuk membantu pengambilan keputusan dalam situasi semi terstruktur dan situasi yang tidak terstruktur, di mana tak seorang pun tahu secara pasti bagaimana keputusan seharusnya dibuat (Kurniasih, 2013).

Sprague dan Watson mendefinisikan SPK sebagai sistem yang memiliki lima karakteristik utama yaitu:

1. Sistem yang berbasis komputer.
2. Dipergunakan untuk membantu para pengambil keputusan.
3. Untuk memecahkan masalah-masalah rumit yang mustahil dilakukan dengan kalkulasi manual.
4. Melalui cara simulasi yang interaktif.
5. Data dan model analisis merupakan komponen utama.

### 2.2.1 Jenis Keputusan

Sistem Pendukung Keputusan adalah sebuah sistem yang mampu memberikan kemampuan, baik kemampuan pemecahan masalah maupun

kemampuan pengkomunikasian untuk masalah semi terstruktur.

Secara khusus, Sistem Pendukung Keputusan adalah 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.

**Keputusan tidak terstruktur** (*unstructured decision*) adalah keputusan yang pengambilan keputusannya harus memberikan penilaian, evaluasi, dan pengertian untuk memecahkan masalahnya. Setiap keputusan ini adalah baru, penting, dan tidak rutin, serta tidak ada pengertian yang dipahami benar atau prosedur yang disetujui bersama dalam pengambilannya.

**Keputusan terstruktur** (*structured decision*), sifatnya berulang dan rutin, dan melibatkan prosedur yang jelas dalam menanganinya, sehingga tidak perlu diperlakukan seakan-akan masih baru. Banyak keputusan memiliki elemen-elemen dari kedua jenis keputusan ini.

**Keputusan semistruktur** (*semistructured decision*), yaitu yang hanya sebagian masalahnya mempunyai jawaban yang jelas tersedia dengan prosedur yang disetujui bersama. Secara umum, keputusan terstruktur lebih umum dijumpai pada tingkat organisasi rendah, sedangkan masalah yang tidak terstruktur lebih umum dijumpai pada tingkat tinggi. (Sari, Indah Kumala dkk., 2009)

### 2.2.2 Tahapan Sistem Pengambilan Keputusan

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

1. Penelusuran (*intelligence*)

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

1. Perancangan (*design*)

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

1. Pemilihan (*choice*)

Yaitu memilih alternatif solusi yang diperkirakan paling sesuai.

1. Implementasi (*implementation*)

Tahap ini merupakan tahap pelaksanaan dari keputusan yang telah diambil.

(Kurniasih, 2013)

### 2.2.3 Karakteristik, Kemampuan, dan Keterbatasan SPK

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

# A. Karakteristik SPK

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

# B. Kemampuan SPK

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

## C. Keterbatasan SPK

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

SPK tidak memiliki kemampuan intuisi seperti yang dimiliki oleh manusia. Karena walau bagaimanapun canggihnya suatu SPK, hanyalah suatu kumpulan perangkat keras, perangakat lunak dan sistem operasi yang tidak dilengkapi dengan kemampuan berpikir (Kurniasih, 2013).

### 2.3 Skala Likert

Rensis Likert telah mengembangkan sebuah skala untuk mengukur sikap masyarakat di tahun 1932 yang sekarang terkenal dengan nama skala Likert. Djaali (2008:28) menjelaskan bahwa skala Likert digunakan untuk mengukur sikap, pendapat, dan persepsi seseorang atau sekelompok orang tentang fenomena sosial. Dengan skala Likert, variabel yang akan diukur dijabarkan menjadi indikator variabel. Kemudian indikator tersebut dijadikan sebagai titik tolak untuk menyusun *item*-*item* instrumen yang dapat berupa pertanyaan atau pernyataan. Jawaban setiap *item* instrumen yang menggunakan Skala Likert mempunyai gradasi dari sangat positif sampai sangat negatif, yang dapat berupa kata-kata antara lain: Sangat Penting (SP), Penting (P), Ragu-ragu (R), Tidak Penting (TP), Sangat Tidak Penting (STP).

Skala Likert adalah suatu skala psikometrik yang umum digunakan dalam kuesioner, dan merupakan skala yang paling banyak digunakan dalam riset berupa survei. Sewaktu menanggapi pertanyaan dalam Skala Likert, responden menentukan tingkat persetujuan mereka terhadap suatu pernyataan dengan memilih salah satu dari pilihan yang tersedia. Ada dua bentuk pertanyaan yang menggunakan Likert yaitu pertanyaan positif untuk mengukur minat positif, dan bentuk pertanyaan negatif untuk mengukur minat negatif. Pertanyaan positif diberi skor 5, 4, 3, 2, dan 1, sedangkan bentuk pertanyaan negatif diberi skor 1, 2, 3, 4, dan 5. Bentuk jawaban Skala Likert terdiri dari sangat setuju, setuju, ragu-ragu, tidak setuju, dan sangat tidak setuju. Biasanya disediakan lima pilihan skala dengan format seperti:

1. Sangat tidak setuju
2. Tidak setuju
3. Netral
4. Setuju
5. Sangat setuju

Penskalaan ini apabila dikaitkan dengan jenis data yang dihasilkan adalah data Ordinal. Selain pilihan dengan lima skala tersebut, kadang digunakan juga skala dengan tujuh atau sembilan tingkat. Suatu studi empiris menemukan bahwa beberapa karakteristik statistik hasil kuesioner dengan berbagai jumlah pilihan tersebut ternyata sangat mirip. Skala Likert merupakan metode skala bipolar yang mengukur baik tanggapan positif ataupun negatif terhadap suatu pernyataan. Empat skala pilihan juga kadang digunakan untuk kuesioner Skala Likert yang memaksa orang memilih salah satu kutub karena pilihan "netral" tak tersedia. Jenis data ada empat NOIR (Nominal, Ordinal, Interval, Rasio) keempat jenis data ini memiliki ciri sebagai berikut:

1. Nominal: Bersifat mengklasifikasikan saja, tanpa ada jenjang di antara klasifikasi. Angka hanya bermakna sebagai variasi jenis tanpa bermakna tingkatan. Misal: laki-laki – perempuan, 1 untuk kode laki-laki dan 2 untuk kode perempuan, angka 1 dan 2 bukan merupakan tingkatan, yang artinya 2 bukan berarti lebih tinggi daripada 1. Data jenis ini belum bisa dilakukan operasi matematis.
2. Ordinal: Bersifat mengklasifikasikan, dan klasifikasi tersebut sudah merupakan tingkatan. Sehingga dengan data ordinal ini angka sudah menunjukkan mana yg lebih besar dan mana yang lebih kecil. Tetapi masing-masing klasifikasi yang berupa tingkatan tersebut tidak memiliki jarak yang sama. Misal : juara dalam perlombaan balap sepeda. Ada juara 1, juara 2, dan juara 3. Angka 1, 2, 3 tersebut sudah memiliki makna tingkatan, bahwa juara 1 lebih cepat daripada juara 2 dan juara 3. Juara 2 lebih cepat daripada juara 3. Juara 1 waktu tempuahnya 5 menit, Juara 2 waktu tempuhnya 7 menit dan juara 3 waktu tempuhnya 12 menit. Yang dimaksud tidak memiliki jarak yang sama adalah antara juara 1 dan 2 selisih waktunya 2 menit, antara juara 2 dan juara 3 selisih waktunya 5 menit. Dengan demikian kita masih belum bisa menggunakan operasi matematis, karena angka 1, 2 dan 3 itu hanya berupa *ranking* saja.
3. Interval: Bersifat mengklasifikasikan, dan klasifikasi tersebut sudah merupakan tingkatan yang masing-masing tingkatan memiliki jarak yang sama. Misal: nomor sepatu. Sepatu dengan nomor 39, 40, 41, 42. Angka nomor sepatu tersebut sudah bermakna tingkatan bahwa nomor 42 lebih tinggi daripada nomor 41 dan seterusnya. Pada data interval masing-masing tingkatan tersebut memiliki jarak yang sama. Sepatu nomor 39 memiliki panjang 30cm, nomor 40 memiliki panjang 31cm, nomor 41 memiliki panjang 32cm, nomor 42 memiliki panjang 33cm. dengan contoh tersebut berarti setiap tingkatan memiliki interval 1cm, interval inilah yang dimaksud dengan jarak yang sama di masing-masing tingkatan. Dengan adanya interval yang diketahui tersebut, bisa dimaknai bahwa nomor sepatu 42 adalah nomor 39 ditambah 3cm, tapi belum bisa dimaknai bahwa nomor 42 adalah nomor 39 dikali 3. Dengan demikian bisa disimpulkan bahwa data interval sudah bisa dikenai operasi matematis penjumlahan dan pengurangan, namun belum bisa dikenai operasi matematis perkalian dan pembagian. Hal ini karena data interval tidak memiliki angka nol mutlak.
4. Rasio: ini adalah data dengan tingkatan yang tertinggi karena telah memiliki angka nol mutlak. Misal ukuran panjang atau tinggi, dan ukuran berat. Berat 0 kg berarti memang tidak ada massa yang ditimbang. Berat 3kg lebih besar daripada berat 2kg, berat 2kg lebih besar daripada berat 1kg. Sehingga berdasarkan contoh tersebut kita bisa memaknai bahwa 3kg adalah 2kg + 1kg atau 3kg adalah 3 x 1kg. Dengan demikian data rasio sudah bisa dikenai semua operasi matematis: +, -, x, dan :.

Skala Likert termasuk data ordinal, karena sangat setuju pasti lebih tinggi daripada yang setuju. Yang setuju pasti lebih tinggi daripada yang netral, yang netral pasti lebih tinggi daripada yang tidak setuju, sedangkan yang tidak setuju pasti lebih tinggi daripada yang sangat tidak setuju. Namun jarak antara sangat setuju ke setuju dan dari setuju ke netral dan seterusnya tentunya tidak sama, oleh karena itu data yang dihasilkan oleh skala Likert adalah data ordinal. Sedangkan cara *scoring* bahwa Sangat setuju 5, setuju 4, netral 3, tidak setuju 2 dan sangat tidak setuju 1 hanya merupakan kode saja untuk mengetahui mana yang lebih tinggi dan mana yang lebih rendah (Nirwarni, 2010).

### 2.4 Metode *Weighted Product* (WP)

*Multi Attribute Decision Making (*MADM*)* adalah suatu metode yang

digunakan untuk mencari alternatif paling optimal dari sejumlah alternatif optimal dengan kriteria tertentu. Inti dari MADM adalah menentukan nilai bobot untuk setiap atribut, kemudian dilanjutkan dengan proses perangkingan yang akan menyeleksi alternatif yang sudah diberikan. *Weighted Product (WP)* merupakan salah satu metode yang digunakan untuk menyelesaikan masalah MADM. *Weighted Product (WP)* adalah suatu metode yang menggunakan perkalian untuk menghubungkan rating atribut, di mana *rating* setiap atribut harus dipangkatkan dulu dengan bobot yang bersangkutan. Proses ini sama halnya dengan proses normalisasi. (Sianturi, 2013)

Metode *Weighted Product* dapat membantu dalam mengambil keputusan pemilihan laptop, akan tetapi perhitungan dengan menggunakan metode *Weighted Product* ini hanya menghasilkan nilai terbesar yang akan terpilih sebagai alternatif yang terbaik. Perhitungan akan sesuai dengan metode ini apabila alternatif yang terpilih memenuhi kriteria yang telah ditentukan. Metode WP ini lebih efisien karena waktu yang dibutuhkan dalam perhitungan lebih singkat. Bobot untuk atribut manfaat berfungsi sebagai pangkat positif dalam proses perkalian, sementara bobot biaya berfungsi sebagai pangkat negatif. (Sari, Indah Kumala dkk., 2009)

Perbaikan bobot untuk *Wj*1 adalah dengan menggunakan rumus

(2.1)

=

∑



Variabel *W* adalah pangkat bernilai positif untuk atribut keuntungan dan bernilai negatif untuk atribut biaya.Preferensi untuk alternatif *Si*diberikan sebagai

berikut:

*Si*=∏ (2.2)



|  |  |
| --- | --- |
| Dengan *i* = 1, 2, …, *m* dan *j* sebagai atribut=1, 2, …, *n*.  Keterangan:  Π : product  *Si* : skor / nilai dari setiap alternatif  *Xij* : nilai alternatif ke- *i* terhadap atribut ke- *j*  *w*j = bobot dari setiap atribut atau kriteria  *n* : Banyaknya kriteria  Untuk mencari alternatif terbaik dilakukan dengan persamaan berikut: |  |
| *Vi*  =  ∏  (  ∗  ) | (2.3) |

di mana :

1. : Preferensi alternatif dianalogikan sebagai vektor *V*

*X* : Nilai Kriteria

1. : Bobot kriteria/subkriteria *i* : Alternatif *j* : Kriteria *n* : Banyaknya kriteria

\* : Banyaknya kriteria yang telah dinilai pada vektor *S*

Nilai *Vi* yang terbesar menyatakan bahwa alternatif *A*i yang terpilih.

Langkah-langkah dalam perhitungan metode WP adalah sebagai berikut:

1. Mengalikan seluruh atribut bagi seluruh alternatif dengan *W* (bobot) sebagai pangkat positif untuk atribut keuntungan dan bernilai negatif untuk atribut biaya. 2. Hasil perkalian dijumlahkan untuk menghasilkan nilai pada setiap alternatif 3. Membagi nilai *V* bagi setiap alternatif dengan nilai total dari semua nilai

alternatif.

4. D*item*ukan urutan alternatif terbaik yang akan menjadi keputusan. (Putra Jaya, 2013).

Adapun kelebihan dan kekurangan dari metode *weighted product* ini adalah sebagai berikut:

a. Kelebihan Metode WP

1. Mempercepat proses perhitungan nilai kriteria dan perangkingan untuk setiap

alternatif.

1. Mempermudah *user* untuk memberikan pembobotan terhadap kriteria yang memiliki nilai yang hampir sama.
2. Dapat digunakan untuk pengambilan keputusan *single* dan keputusan

*multidimensional*.

1. Metode ini digunakan untuk menyelesaikan pengambilan keputusan secara praktis, karena konsepnya sederhana dan mudah dipahami.

b. Kelemahan Metode WP

Adapun kelemahan dari metode *weighted product* ini adalah sebagai berikut:

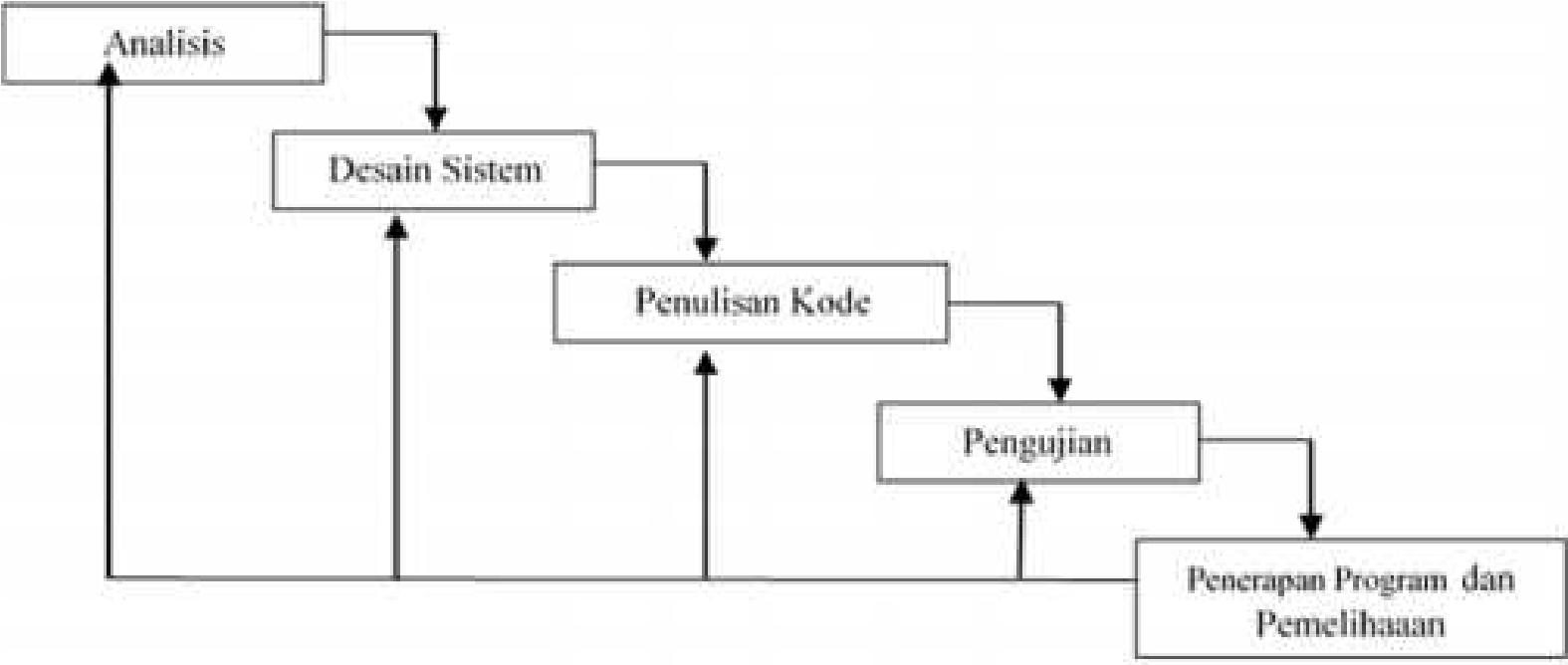
1. Tidak banyak *user* yang menggunakan metode ini dalam pengambilan keputusan.
2. Metode ini hanya metode matematis tanpa ada pengujian secara statistik sehingga tidak ada batas kepercayaan dari kebenaran model yang terbentuk.

### 2.5 Metode Pengembangan Sistem

#### 2.5.1 Metode *Waterfall*

Metode *waterfall* merupakan metode yang sering digunakan oleh penganalisis sistem pada umumnya. Inti dari metode *waterfall* adalah pengerjaan dari suatu sistem dilakukan secara berurutan atau secara linear. Jadi jika langkah ke-1 belum dikerjakan, maka langkah 2 tidak dapat dikerjakan. Jika langkah ke-2 belum dikerjakan maka langkah ke-3 juga tidak dapat dikerjakan, begitu seterusnya.

Secara otomatis langkah ke-3 akan bisa dilakukan jika langkah ke-1 dan ke2 sudah dilakukan, seperti Gambar 2.1:



# Gambar 2.1 Model *waterfall* menurut Roger S. Pressman (Marselia, 2012 )

Berikut adalah penjelasan dari tahap-tahap yang dilakukan di dalam model ini menurut Pressman:

1. Analisis Kebutuhan

Langkah ini merupakan analisis terhadap kebutuhan sistem. Pengumpulan data dalam tahap ini bisa melakukan sebuah penelitian, wawancara atau studi literatur. Sistem analisis akan menggali informasi sebanyak-banyaknya dari *user* sehingga akan tercipta sebuah sistem komputer yang bisa melakukan tugastugas yang diinginkan oleh *user* tersebut. Tahapan ini akan menghasilkan dokumen *user requirment* atau bisa dikatakan sebagai data yang berhubungan dengan keinginan *user* dalam pembuatan sistem. Dokumen ini lah yang akan menjadi acuan sistem analis untuk menerjemahkan ke dalam bahasa pemrogram.

1. Desain Sistem

Tahapan di mana dilakukan penuangan pikiran dan perancangan sistem terhadap solusi dari permasalahan yang ada dengan menggunakan perangkat pemodelan sistem seperti diagram alir data (*data flow diagram*), diagram hubungan entitas (*entity relationship diagram*) serta struktur dan bahasan data.

1. Penulisan Kode Program

Penulisan kode program atau *coding* merupakan

penerjemahan *design* dalam bahasa yang bisa dikenali oleh komputer. Dilakukan oleh *programmer* yang akan menerjemahkan transaksi yang diminta oleh *user.* Tahapan ini lah yang merupakan tahapan secara nyata dalam mengerjakan suatu sistem. Dalam artian penggunaan komputer akan dimaksimalkan dalam tahapan ini. Setelah pengkodean selesai maka akan dilakukan *testing* terhadap sistem yang telah dibuat tadi. Tujuan *testing* adalah menemukan kesalahan-kesalahan terhadap sistem tersebut dan kemudian bisa diperbaiki.

1. Pengujian Program

Tahapan akhir di mana sistem yang baru diuji kemampuan dan keefektifannya sehingga didapatkan kekurangan dan kelemahan sistem yang kemudian dilakukan pengkajian ulang dan perbaikan terhadap aplikasi menjadi lebih baik dan sempurna.

1. Penerapan Program dan Pemeliharaan

Perangkat lunak yang sudah disampaikan kepada pelanggan pasti akan mengalami perubahan. Perubahan tersebut bisa karena mengalami kesalahan karena perangkat lunak harus menyesuaikan dengan lingkungan (periperal atau sistem operasi baru) baru, atau karena pelanggan membutuhkan perkembangan fungsional (Marselia, 2012).

## 2.6 UML (*Unified Modeling Language*)

UML (*Unified Modeling Language*) adalah metode pemodelan secara visual sebagai sarana untuk merancang dan atau membuat *software* berorientasi objek. Karena UML ini merupakan bahasa visual untuk pemodelan bahasa berorientasi objek, maka semua elemen dan diagram berbasiskan pada paradigma *object oriented*.

UML adalah salah satu *tool*/model untuk merancang pengembangan *software* yang berbasis *object oriented*. UML sendiri juga memberikan standar penulisan sebuah sistem *blue* print, yang meliputi konsep bisnis proses, penulisan kelas-kelas dalam bahasa program yang spesifik, skema *database*, dan komponen- komponen yang diperlukan dalam sistem *software*.

UML adalah sebuah bahasa standar untuk pengembangan sebuah *software* yang dapat menyampaikan bagaimana membuat dan membentuk model-model, tetapi tidak menyampaikan apa dan kapan model yang seharusnya dibuat yang merupakan salah satu proses implementasi pengembangan *software*. UML tidak hanya merupakan sebuah bahasa pemograman *visual* saja, namun juga dapat secara langsung dihubungkan ke berbagai bahasa pemograman, seperti JAVA, C++, *Visual Basic*, atau bahkan dihubungkan secara langsung ke dalam sebuah *objectoriented database*. Begitu juga mengenai pendokumentasian dapat dilakukan seperti *requirements*, arsitektur, *design*, *source code*, *project plan*, *tests*, dan *prototypes*.

UML sendiri terdiri atas pengelompokkan diagram-diagram sistem menurut aspek atau sudut pandang tertentu. Diagram adalah yang menggambarkan permasalahan maupun solusi dari permasalahan suatu model. UML mempunyai 9 diagram, yaitu;

* Diagram *Use Case*
* Diagram *Class*

##  Diagram *Package*  Diagram *Sequence*  Diagram *Collaboration*  Diagram *StateChart*  Diagram *Activity*  Diagram *Deployment*

UML yang akan digunakan yaitu Diagram *Use Case*, Diagram *Sequence*, dan Diagram *Activity*, dan diagram *Class* karena pada umumnya pembuatan sistem hanya menggunakan keempat diagram tersebut.

1. Diagram *Use Case*

*Use case diagram* menggambarkan fungsionalitas yang diharapkan dari sebuah sistem yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Menggambarkan kebutuhan sistem dari sudut pandang *user*. Memfokuskan pada proses komputerisasi (*automated processes*).

Menggambarkan hubungan antara *use case* dan *actor use case,* menggambarkan proses sistem (kebutuhan sistem dari sudut pandang *user*).

**Tabel 2.1 Simbol Diagram *Use Case***

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Gambar | Nama | Keterangan |
| 1 |  | *Actor* | Menspesifikasikan himpunan peran yang pengguna main kan ketika berinteraksi dengan *use case*. |
| 2 |  | *Dependency* | Hubungan di mana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempengaruhi elemen yang bergantung padanya  (*independent*). |
| 3 |  | *Generalization* | Hubungan di mana objek anak  (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya, objek induk (*ancestor*). |
| 4 |  | *Include* | Menspesifikasikan bahwa *use case*  sumber secara *eksplisit*. |
| 5 |  | *Extend* | Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan. |
| 6 |  | *Association* | Apa yang menghubungkan antara objek satu dengan objek lainnya. |
| 7 |  | *System* | Menspesifikasikan paket yang  menampilkan sistem secara terbatas. |
| 8 |  | *Use Case* | De*script*si dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu *actor* |

**Tabel 2.1 Lanjutan**

|  |  |  |  |
| --- | --- | --- | --- |
| 9 |  | *Collaboration* | Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan prilaku yang lebih besar dari jumlah dan elemenelemennya (sinergi). |
| 10 |  | *Note* | Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi. |

1. Diagram *Sequence*

**Tabel 2.2 Simbol Diagram *Sequence***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Gambar | | Nama | Keterangan |  |
|  | | *LifeLine* | Objek *entity*, antarmuka yang saling berinteraksi. |
|  | | *Message* | Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi |
|  | *Sequence* diagram menggambarkan interaksi antar objek di dalam dan di sekitar | | | |

sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence* diagram terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence* diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan. Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/*metoda* dari *class. Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message*.

1. Diagram *Activity*

*Activity diagram* memodelkan *workflow* proses bisnis dan urutan aktivitas dalam sebuah proses. Diagram ini sangat mirip dengan *flowchart* karena

memodelkan *workflow* dari satu aktivitas ke aktivitas lainnya atau dari aktivitas ke status. Menguntungkan untuk membuat *activity diagram* pada awal pemodelan proses untuk membantu memahami keseluruhan proses. *Activity diagram* juga bermanfaat untuk menggambarkan *parallel behaviour* atau menggambarkan

interaksi antara beberapa *use case*.

**Tabel 2.3 Simbol Diagram *Activity***

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **GAMBAR** | **NAMA** | **KETERANGAN** |
| **1** |  | *Activity* | Memperlihatkan bagaimana masing-masing kelas antarmuka  saling berinteraksi satu sama lain |
| **2** |  | *Action* | *State* dari sistem yang mencerminkan eksekusi dari suatu aksi |
| **3** |  | *Initial Node* | Bagaimana objek dibentuk atau diawali. |
| **4** |  | *Activity Final*  *Node* | Bagaimana objek dibentuk dan dihancurkan |
| **5** |  | *Fork Node* | Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran |

Semakin kompleks bentukan sistem yang akan dibuat, maka semakin sulit komunikasi antara orang-orang yang saling terkait dalam pembuatan dan pengembangan *software* yang akan dibuat. Pada masa lalu, UML mempunyai peranan sebagai *software blueprint* (gambaran) *language* untuk analis sistem, *designer*, dan *programmer*. Sedangkan pada saat ini, merupakan bagian dari *software trade* (bisnis perangkat lunak). UML memberikan jalur komunikasi dari sistem analis kemudian *designer*, lalu *programmer* mengenai rancangan *software* yang akan dikerjakan.

Model adalah gambaran abstrak dari suatu dasar masalah dan dunia nyata atau tempat di mana masalah itu timbul, bisa disebut dengan domain. Model mengandung obyek-obyek yang beraktifitas dengan saling mengirimkan *messages* (pesan-pesan). Obyek mempunyai sesuatu yang diketahui (atribut) dan sesuatu yang dilakukan (*behaviors* atau *operations*). Atribut hanya berlaku dalam ruang lingkup obyek itu sendiri (*state*). Lalu “*blue print*” dari suatu obyek adalah *classes* (kelas). Obyek merupakan bagian-bagian dari kelas.

### *4. Class Diagram*

*Class diagram* atau diagram kelas yaitu salah satu jenis diagram pada UML yang digunakan untuk menampilkan kelas-kelas yang ada pada suatu sistem yang nantinya akan digunakan. Kelas memiliki atribut dan metode atau operasi. Atribut merupakan variable-variabel yang dimiliki oleh suatu kelas. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas. Diagram kelas dibuat agar program atau *programmer* membuat kelas-kelas sesuai dengan rancangan di dalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak sinkron. Susunan struktur kelas yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas berikut:

* Kelas *main* adalah kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.
* Kelas yang menangani tampilan sistem (*view*) adalah kelas yang mendefinisikan dan mengatur tampilan ke pemakai.
* Kelas yang diambil dari pendefinisian *use case (controller)* adalah kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *use case*, kelas ini biasanya disebut dengan kelas proses yang menangani proses bisnis pada perangkat lunak.
* Kelas yang diambil dari pendefinisian data (*model*) adalah kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan di basis data. Berikut adalah simbol-simbol yang ada pada diagram kelas:

# Tabel 2.4 simbol-simbol diagram kelas

|  |  |  |
| --- | --- | --- |
| **Nama Komponen** | **Keterangan** | **Simbol** |
| ***Class*** | *Class* adalah blok-blok pembangun pada pemrograman berorientasi obyek. Sebuah *class* digambarkan sebagai sebuah kotak yang terbagi atas 3 bagian. Bagian atas adalah bagian nama dari *class*. Bagian tengah mendefinisikan property/atribut *class*. Bagian akhir mendefinisikan *method- method* dari sebuah *class*. | |  | | --- | | Nama kelas | | + atribut  + atribut  + atribut | | + *method*  + *method* | |
| ***Association*** | Sebuah asosiasi merupakan sebuah relationship paling umum antara 2 *class* dan dilambangkan oleh sebuah garis yang menghubungkan antara 2 *class*. Garis ini bisa melambangkan tipe-tipe *relationship* dan juga dapat menampilkan hukumhukum multiplisitas pada sebuah *relationship.*  (Contoh: *One-to-one, one-to-many, manyto-many*). |  |
| ***Composition*** | Jika sebuah *class* tidak bisa berdiri sendiri dan harus merupakan bagian dari *class* yang lain, maka *class* tersebut memiliki relasi *Composition* terhadap *class* tempat dia bergantung tersebut. Sebuah *relationship composition* digambarkan sebagai garis dengan ujung berbentuk jajaran genjang berisi/solid. |  |
| ***Dependency*** | Kadangkala sebuah *class* menggunakan *class* yang lain. Hal ini disebut *dependency.*  Umumnya penggunaan *dependency* digunakan untuk menunjukkan operasi pada suatu *class* yang menggunakan *class* yang lain. Sebuah *dependency* dilambangkan sebagai sebuah panah bertitik-titik. |  |

## 2.7 *Flowchart*

### 2.7.1 Pengertian Dasar *Flowchart*

*Flowchart* adalah bagan-bagan yang mempunyai arus yang menggambarkan langkah-langkah penyelesaian suatu masalah. *Flowchart* merupakan cara penyajian dari suatu algoritma.

Tujuan Membuat *Flowchart*:

* Menggambarkan suatu tahapan penyelesaian masalah
* Secara sederhana, terurai, rapi dan jelas
* Menggunakan simbol-simbol standar

Dalam penulisan *flowchart* dikenal dua model, yaitu Sistem *Flowchart* dan Program *Flowchart*.

## 1. Sistem *Flowchart*

Sistem *Flowchart* adalah bagan yang memperlihatkan urutan *prosedure* dan proses dari beberapa *file* di dalam media tertentu.

Melalui *flowchart* ini terlihat jenis media penyimpanan yang dipakai dalam pengolahan data.

Selain itu juga menggambarkan *file* yang dipakai sebagai *input* dan *output*.

Tidak digunakan untuk menggambarkan urutan langkah untuk

memecahkan masalah, hanya untuk menggambarkan prosedur dalam sistem yang

dibentuk.

## 2. Program *Flowchart*

Program *flowchart* adalah bagan yang memperlihatkan urutan dan hubungan proses dalam suatu program.

Dua jenis metode penggambaran program *flowchart*:

1. *Conceptual flowchart*, menggambarkan alur pemecahan masalah secara

global.

1. *Detail flowchart*, menggambarkan alur pemecahan masalah secara rinci.

### 2.7.2 Bagan *Flowchart*

*Flowchart* adalah serangkaian bagan-bagan yang menggambarkan alir program. *Flowchart* atau diagram alir memiliki bagan-bagan yang melambangkan fungsi tertentu. Bagan, nama dan fungsinya seperti yang disajikan pada Tabel 2.5 tabel bagan *flowchart*.

**Tabel 2.5 Tabel bagan *Flowchart***

|  |  |  |
| --- | --- | --- |
| Bagan | Nama | Fungsi |
|  | *Terminator* | Awal atau akhir program |
|  | *Flow* | Arah aliran program |
|  | *Preparation* | Inisialisasi/pemberian nilai awal |
|  | *Process* | Proses/pengolahan data |
|  | *Input/output* data | *Input/output* data |
|  | Sub program | Sub program |
|  | *Decision* | Seleksi atau kondisi |
|  | *On page connector* | Penghubung bagianbagian *flowchart* pada halaman yang sama |
|  | *Off page connector* | Penghubung bagianbagian *flowchart* pada  halaman yang berbeda |
|  | *Comment* | Tempat komentar tentang suatu proses |

### 2.8 *Database* (Basis Data)

#### 2.8.1 Pengertian *Database*

*Database* atau Basis Data adalah sekumpulan data yang saling terhubung satu dengan yang lainnya atau sekumpulan tabel yang saling terhubung satu dengan yang lainnya. Fungsi dari *database* adalah menyimpan suatu data pada tabel-tabel dan dikumpulkan menjadi satu dengan *database*. *Database* juga bisa diumpamakan sebagai sebuah rumah dengan beberapa kamar-kamar dan sebuah *property* seperti almari, meja belajar, tempat tidur, itu bisa disebut dengan data *query-*nya. Ada beberapa bagian bagian dari *database* yaitu:

# A). Komponen *Database*

Komponen yang terdapat pada suatu *database* antara lain:

## 1). *Tabel*

Sebuah komponen yang digunakan untuk menyimpan suatu data yang telah di akses dan dimasukkan ke dalamnya.

## 2). *Record*

Isi atau data dari tabel tersebut yang telah dikelola. *Record* dapat mempunyai beberapa macam data. Data bervariasi tersebut disimpan ke dalam tabel dan itulah yang disebut *record*.

## 3). *Field*

Pemberian identitas suatu data di mana data tersebut akan diletakkan. Sesuai dengan pengelompokan datanya.

# B) Struktur *Database*

Struktur *database* adalah suatu pengaturan *field-field* pada suatu tabel pada *database*. Beberapa struktur *database* sebagai berikut:

1). Nama *Field*

Digunakan sebagai suatu pemberian identitas atau memberi keterangan pada *field*.

2). *Type* Data

Pemberian suatu tipe pada *field* sesuai dengan identitas yang telah diberikan.

3). Ukuran Data

Pemberian suatu panjang atau banyak data yang telah di masukan.

4). Keterangan

Memberikan suatu keterangan atau de*script*si pada sebuah *field*.

## 2.9 *Website*

*Website* adalah kumpulan dari halaman - halaman situs, yang terangkum dalam sebuah domain atau subdomain, yang tempatnya berada di dalam *World Wide Web* (WWW) di dalam internet. Sebuah halaman web biasanya berupa dokumen yang ditulis dalam format HTML (*Hyper Text Markup Language*), yang selalu bisa diakses melalui HTTP, yaitu sebuah protokol yang menyampaikan informasi dari *server website* untuk ditampilkan kepada para pemakai melalui *web browser*. *Website* atau situs dapat juga diartikan sebagai kumpulan halaman yang menampilkan informasi data teks, data gambar diam atau gerak, data animasi, suara, video dan atau gabungan dari semuanya, baik yang bersifat statis maupun dinamis yang membentuk satu rangkaian bangunan yang saling terkait dimana masingmasing dihubungkan dengan jaringan-jaringan halaman (*hyperlink*).

### 2.9.1 Jenis-Jenis *Website*

Secara garis besar, *website* bisa digolongkan menjadi 3 bagian yaitu:

1. *Website* Statis

*Website* Statis adalah web yang mempunyai halaman tidak berubah. Artinya adalah untuk melakukan perubahan pada suatu halaman dilakukan secara manual dengan mengedit *code* yang menjadi struktur dari *website* tersebut.

1. *Website* Dinamis

*Website* Dinamis merupakan *website* yang secara struktur diperuntukan untuk *update* sesering mungkin. Biasanya selain halaman utama yang bisa diakses oleh *user* pada umumnya, juga disediakan halaman *backend* untuk mengedit kontent dari *website*. Contoh umum mengenai *website* dinamis adalah web berita atau web portal yang didalamnya terdapat fasilitas berita, polling dan sebagainya.

1. *Website* Interaktif

Website Interaktif adalah web yang saat ini memang sedang *booming*. Salah satu contoh *website* interaktif adalah blog dan forum. Di *website* ini *user* bisa berinteraksi dan beradu argument mengenai apa yang terjadi. Biasanya *website* seperti memiliki moderator untuk mengatur supaya topik yang diperbincangkan tidak keluar jalur.

## 2.10 Perangkat Lunak Pendukung

Adapun perangkat lunak yang digunakan untuk membangun sistem ini yaitu *NetBeans*, *MySQL* dan *Xampp Version 1.7.* sebagai pengelola *Script PHP* serta mendukung *database server* pada *MySQL*. Berikut ini penjelasan dari kedua perangkat lunak pendukung tersebut:

### 2.10.1 Bahasa Pemrograman *Hypertext Preprocessing* (PHP)

PHP merupakan bahasa standar yang digunakan dalam dunia *website*. PHP adalah bahasa pemograman yang berbentuk *script* yang diletakan di dalam *server* web. PHP diciptakan dari ide Rasmus Lerdof untuk kebutuhan pribadinya. *Script* tersebut sebenarnya dimaksudkan untuk digunakan sebagai keperluan membuat *website* pribadi. Akan tetapi kemudian dikembangkan lagi sehingga menjadi bahasa yang disebut *“Personal Home Page”.* Inilah awal mula munculnya PHP

sampai saat ini.

PHP dirancang untuk membentuk web dinamis. Artinya, PHP dapat membentuk suatu tampilan berdasarkan permintaan terkini. Pada prinsipnya, PHP mempunyai fungsi yang sama dengan *script* seperti ASP (*Actives Server Page*), *Cold Fusion*, ataupun *Perl*.

### 2.10.2 *NetBeans*

*NetBeans* adalah sebuah aplikasi *Integrated Development Environment*

(IDE) yang berbasiskan Java dari *Sun Microsystems* yang berjalan di atas *swing. Swing* merupakan sebuah teknologi Java untuk pengembangan aplikasi *dekstop* yang dapat berjalan pada berbagai macam *platform* seperti Windows, Linux, Mac OS X dan Solaris. Sebuah IDE merupakan lingkup pemrograman yang diintegrasikan ke dalam suatu aplikasi perangkat lunak yang menyediakan *Graphic User Interface* (GUI), suatu kode *editor* atau *text*, suatu *compiler* dan suatu *debugger*.

*NetBeans* juga dapat digunakan *progammer* untuk menulis, meng-*compile*, mencari kesalahan dan menyebarkan program *NetBeans* yang ditulis dalam bahasa pemrograman Java namun selain itu dapat juga mendukung bahasa pemrograman lainnya dan program ini pun bebas untuk digunakan dan untuk membuat *professional dekstop, enterprise, web, and mobile applications* dengan Java *language*, C/C++, dan bahkan *dynamic languages* seperti PHP, *JavaScript*, *Groovy*, dan *Ruby*.

*NetBeans* merupakan sebuah proyek kode terbuka yang sukses dengan pengguna yang sangat luas, komunitas yang terus tumbuh, dan memiliki hampir 100 mitra. *Sun Microsystems* mendirikan proyek kode terbuka *NetBeans* pada bulan Juni 2000 dan terus menjadi sponsor utama. Dan saat ini pun *NetBeans* memiliki 2 produk yaitu *Platform NetBeans* dan *NetBeans* IDE. *Platform NetBeans* merupakan *framework* yang dapat digunakan kembali (*reusable*) untuk menyederhanakan pengembangan aplikasi *desktop* dan *Platform NetBeans* juga menawarkan layananlayanan yang umum bagi aplikasi *dekstop*, mengizinkan pengembang untuk fokus ke logika yang spesifik terhadap aplikasi.

Fitur-fitur yang terdapat dalam *NetBeans* antara lain:

1. *Smart Code Completion*: untuk mengusulkan nama variabel dari suatu tipe, melengkapi *keyword* dan mengusulkan tipe parameter dari sebuah *method*.
2. *Bookmarking*: fitur yang digunakan untuk menandai baris yang suatu saat akan di modifikasi.
3. *Go to commands*: fitur yang digunakan untuk *jump* ke deklarasi variabel, *source code* atau *file* yang ada pada *project* yang sama.
4. *Code generator*: jika menggunakan fitur ini, maka dapat meng-*generate constructor*, *setter* and *getter method* dan yang lainnya.
5. *Error stripe*: fitur yang akan menandai baris yang *error* dengan memberi *highlight* merah.

### 2.10.3 Xampp

XAMPP merupakan paket PHP yang berbasis *open source* yang dikembangkan oleh sebuah komunitas *Open Source*. Dengan menggunakan XAMPP tidak dibingungkan dengan penginstalan program-program lain, karena semua kebutuhan terlah tersedia oleh XAMPP. Yang terdapat pada XAMPP di antaranya: *Apache, MySQL, PHP, FilZilla FTP Server, PHPmyAdmin* dan lain-

lain.

Fungsi XAMPP adalah sebagai server yang berdiri sendiri (*localhost*), yang terdiri atas program *Apache* HTTP *Server*, *MySQL database,* dan penerjemah bahasa yang ditulis dengan bahasa pemrograman PHP dan *Perl*. XAMPP adalah perangkat lunak bebas, yang mendukung banyak sistem operasi, merupakan kompilasi dari beberapa program. Fungsinya adalah sebagai *server* yang berdiri sendiri (*localhost*), yang terdiri atas program *Apache* HTTP *Server*, MySQL *database*, dan penerjemah bahasa yang ditulis dengan bahasa pemrograman PHP dan Perl. Nama XAMPP merupakan singkatan dari X (empat sistem operasi apapun), Apache, MySQL, PHP dan Perl. Program ini tersedia dalam GNU *General Public License* dan bebas, merupakan *web server* yang mudah digunakan yang dapat melayani tampilan halaman web yang dinamis.

XAMPP adalah singkatan yang masing-masing hurufnya adalah:

1. X: Program ini dapat dijalankan dibanyak sistem operasi, seperti Windows, Linux, Mac OS, dan Solaris.
2. A: *Apache*, merupakan aplikasi *web server*. Tugas utama Apache adalah menghasilkan halaman *web* yang benar kepada *user* berdasarkan kode PHP yang dituliskan oleh pembuat halaman *web*.
3. M: MySQL, merupakan aplikasi *database server*. Perkembangannya disebut SQL yang merupakan kepanjangan dari *Structured Query Language*. SQL merupakan bahasa terstruktur yang digunakan untuk mengolah *database*. MySQL dapat digunakan untuk membuat dan mengelola *database* beserta isinya. *User* dapat memanfaatkan MySQL untuk menambahkan, mengubah, dan menghapus data yang berada dalam *database*.
4. P: PHP, bahasa pemrograman web. Bahasa pemrograman PHP merupakanbahasa pemrograman untuk membuat web yang bersifat *server-side scripting*.
5. P: Perl, bahasa pemrograman.