**BAB II**

**LANDASAN TEORI**

* 1. **Pengertian Sampah**

Menurut *World Health Organization* (WHO), sampah adalah sesuatu yang tidak digunakan, tidak dipakai, tidak disenangi atau sesuatu yang dibuang berasal dari kegiatan manusia dan tidak terjadi dengan sendirinya.

Sampah dalam ilmu kesehatan lingkungan sebenarnya hanya sebagian dari benda atau hal-hal yang dipandang tidak digunakan, tidak dipakai, tidak disenangi, atau harus dibuang, sedemikian rupa sehingga tidak sampai mengganggu kelangsungan hidup. Dari segi ini dapat disimpulkan bahwa yang dimaksud dengan sampah ialah sebagian dari sesuatu yang tidak dipakai, disenangi atau sesuatu yang harus dibuang, yang umumnya berasal dari kegiatan yang dilakukan oleh manusia (termasuk kegiatan industri), tetapi yang bukan biologis (karena sampah masyarakat tidak termasuk didalamnya) dan umumnya bersifat padat (karena air bekas tidak termasuk didalamnya.Sampah merupakan material sisa yang tidak diinginkan setelah berakhirnya suatu proses. Sampah merupakan konsep buatan manusia, dalam proses-proses alam tidak ada sampah, yang ada hanya produk-produk yang tak bergerak. Sampah dapat berada pada setiap fase materi: padat, cair, atau gas. Ketika dilepaskan dalam dua fase yang disebutkan terakhir, terutama gas, sampah dapat dikatakan sebagai emisi. Emisi biasa dikaitkan dengan polusi. Dalam kehidupan manusia, sampah dalam jumlah besar datang dari aktivitas industri (dikenal juga dengan sebutan limbah), misalnya pertambangan, manufaktur, dan konsumsi. Hampir semua produk industri akan menjadi sampah pada suatu waktu, dengan jumlah sampah yang kira-kira mirip dengan jumlah konsumsi (Tobing, 2005).

* 1. **Tempat Pembuangan Sampah**

**Jenis – Jenis Sampah**

1. **Sampah Rumah Tangga**
   1. Sampah basah

Sampah jenis ini dapat diurai (*degradable*) atau biasa dikatakan membusuk. Contohnya ialah sisa makanan, sayuran, potongan hewan, daun kering dan semua materi yang berasal dari makhluk hidup.

* 1. Sampah kering

Sampah yang terdiri dari logam seperti besi tua, kaleng bekas dan sampah kierng nonlogam seperti kayu, kertas, kaca, keramik, batu-batuan dan sisa kain.

* 1. Sampah lembut

Contoh sampah ini adalah debu dari penyapuan lantai rumah, gedung, penggergajian kayu dan abu dari rokok atau pembakaran kayu.

* 1. Sampah besar

Sampah yang terdiri dari buangan rumah tangga yang besar-besar seperti meja, kursi, kulkas, televisi, radio dan peralatan dapur (Tobing, 2005).

1. **Sampah Komersial**

Sampah yang berasal dari kegiatan komersial seperti pasar, pertokoan, rumah makan, tempat hiburan, penginapan, bengkel dan kios. Demikian pula dari institusi seperti perkantoran, tempat pendidikan, tempat ibadah dan lembaga-lembaga nonkomersial lainnya.

1. **Sampah Bangunan**

Sampah yang berasal dari kegiatan pembangunan termasuk pemugaran dan pembongkaran suatu bangunan seperti semen, kayu, batu-bata dan genting.

1. **Sampah Fasilitas Umum**

Sampah ini berasal dari pembersihan dan penyapuan jalan, trotoar, taman, lapangan, tempat rekreasi dan fasilitas umum lainnya. Contohnya ialah daun, ranting, kertas pembungkus, plastik dan debu.

* 1. **Sistem Penunjang Keputusan**

Sistem Penunjang Keputusan (SPK) adalah bagian dari sistem informasi berbasis komputer termasuk sistem berbasis pengetahuan atau manajemen pengetahuan yang dipakai untuk menunjang pengambilan keputusan dalam suatu organisasi atau perusahaan. Dapat juga dikatakan sebagai sistem komputer yang mengolah data menjadi informasi untuk mengambil keputusan dari masalah semi terstruktur yang spesifik.

Sistem Penunjang Keputusan adalah suatu sistem informasi yang ditujukan untuk membantu pengguna dalam mengambil keputusan yang berkaitan dengan persoalan yang bersifat semi terstruktur, sistem ini memiliki fasilitas untuk menghasilkan berbagai alternatif yang secara interaktif digunakan oleh pengguna. Bagaimanapun juga sistem penunjang keputusan tidak ditekankan untuk membuat keputusan, pengolahan informasi atau data yang diperlukan dalam proses pengambilan keputusan, sistem hanya berfungsi sebagai alat bantu pengguna.

Pembuat keputusan diperlukan pada semua tahap kegiatan administrasi dan manajemen. Misalnya dalam tahap perancangan diperlukan banyak kegiatan pembuatan keputusan sepanjang proses perancangan tersebut. Keputusan-keputusan yang dibuat dalam proses perancangan ditujukan kepada pemilihan alternatif program dan prioritasnya.

Pembuatan keputusan tersebut mencakup kegiatan identifikasi masalah, perumusan dan pemilihan alternatif keputusan berdasarkan perhitungan konsekuensi dan berbagai dampak yang timbul. Begitu juga dalam operasional suatu organisasi, para pengguna harus membuat banyak keputusan rutin dalam rangka mengendalikan usaha sesuai dengan rencana dan kondisi yang berlaku. Sedangkan dalam tahap pengawasan yang mencangkup pemantauan, pemeriksa dan penilaian (evaluasi) terhadap hasil pelaksaan kerja juga banyak keputusan dibuat dalam rangka koreksi terhadap penyimpangan yang terjadi agar hasil yang diperoleh lebih sesuai dengan sasaran mutu, waktu dan penggunaan sumberdaya yang efisien (Turban, 2005).

* 1. **Karakteristik dan Kemampuan Sistem Penunjang Keputusan**

Ada beberapa karakteristik dari SPK, di antarannya adalah sebagai berikut (Turban, 2005):

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 data eksternal dan internal

6. Menggunakan beberapa model kuantitatif

Adapun kemampuan yang harus dimiliki oleh sebuah sistem penunjang keputusan, diantaranya adalah sebagai berikut:

1. Menunjang pembuatan keputusan manajemen dalam menangani masalah semi terstruktur 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 dan perorangan.
4. Menunjang pembuatan keputusan yang saling bergantungan dan berurutan.
5. Menunjang tahap-tahap pembuatan keputusan antara lain *intelligence* (perancangan), *design* (rancangan*), choice* (pilihan) dan *implementation* (implementasi).
6. Menunjang berbagai bentuk proses pembuatan keputusan dan jenis keputusan.
7. Kemampuan untuk melakukan adaptasi setiap saat dan bersifat fleksibel.
8. Kemudahan melakukan interaksi sistem.
9. Meningkatkan efektivitas dalam pembuatan keputusan daripada efisiensi.
10. Mudah dikembangkan oleh pengguna akhir.

Disamping berbagai kemampuan dan karakteristik seperti yang telah dikemukakan , sistem penunjang keputusan juga memiliki keterbatasan antara lain:

1. Ada beberapa kemampuan manajemen dan bakat manusia yang tidak dapat dimodelkan, sehingga model yang ada dalam sistem tidak semuanya mencerminkan persoalan yang sebenarnya.
2. Kemampuan suatu sistem penunjang keputusan terbatas pada pengetahuan dasar serta model dasar yang dimilikinya.
3. Proses-proses yang dapat dilakukan oleh sistem penunjang keputusan biasanya juga tergantung pada kemampuan perangkat lunak yang digunakan.
4. Sistem penunjang keputusan tidak memiliki intuisi seperti yang dimiliki oleh manusia, karena sistem Penunjang keputusan hanya suatu kumpulan perangkat keras, perangkat lunak dan sistem operasi yang tidak dilengkapi oleh kemampuan berpikir.
   1. **Keuntungan Sistem Penunjang Keputusan**

Beberapa keuntungan penggunaan SPK antara lain adalah sebagai berikut (Turban, 2005):

1. Mampu mendukung pencarian solusi dari berbagai permasalahan yang kompleks.
2. Dapat merespon dengan cepat pada situasi yang tidak diharapkan dalam kondisi yang berubah-ubah.
3. Mampu menerapkan berbagai strategi yang berbeda pada konfigurasi berbeda secara cepat dan tepat.
4. Sebagai fasilitator dalam komunikasi.
5. Meningkatkan kontrol manajemen dan kinerja.
6. Menghemat biaya dan sumber daya manusia (SDM).
7. Menghemat waktu karena keputusan dapat diambil dengan cepat.
8. Meningkatkan efektivitas manajerial, menjadikan manajer dapat bekerja lebih singkat dan dengan sedikit usaha.
   1. **Fase Pengambilan Keputusan**

Di dalam sistem penunjang keputusan tidak terlepas dari proses pengambilan keputusan itu sendiri. Pada dasarnya, proses pengambilan keputusan terdiri dari 4 fase proses yaitu(Turban, 2005):

1. *Intelligence/*Inteligensi

Intelegensi dalam pengambilan keputusan meliputi pemindaian *(scanning)* lingkungan. Inteligensi mencakup berbagai aktivitas yang menekankan identifikasi situasi atau peluang – peluang masalah. Fase inteligensi dimulai dengan identifikasi terhadap tujuan dan sasaran organisasional yang berkaitan dengan isu yang terkait dan menentukan apakah tujuan tersebut telah terpenuhi. Pada fase pertama ini, seseorang berusaha menentukan apakah ada suatu masalah, mengidentifikasi gejala-gejalanya, menentukan keluasanya, dan mendefinisikan secara eksplisit.

1. *Design/* Desain

Desainyaitu menemukan, mengembangkan, dan menganalisis materi-materi yang akan dikerjakan.

1. *Choice/* Pilihan

Pilihanyaitu pemilihan materi-materi yang tersedia, materi mana yang akan dikerjakan.

1. *Implementation/* Implementasi

Implementasi merupakan fase penerapan materi yang telah dipilih pada fase sebelumnya.

* 1. **Proses Dalam Sistem Penunjang Keputusan**

Proses-proses yang terjadi pada kerangka kerja sistem penunjang keputusan dibagi menjadi(Turban, 2005):

1. Terstruktur, mengacu pada permasalahan rutin dan berulang untuk solusi standar yang ada.
2. Tak terstruktur, adalah permasalahan kompleks di mana tidak ada solusi serta merta. Masalah yang tidak terstruktur adalah tidak adanya 3 fase proses yang terstruktur.
3. Semi terstruktur, terdapat beberapa keputusan terstruktur, tetapi tidak semuanya dari fase- fase yang ada.

Secara umum, Sistem Penunjang Keputusan (SPK) adalah sistem berbasis komputer yang interaktif, yang membantu pengambil keputusan memanfaatkan data dan model untuk menyelesaikan masalah-masalah yang tak terstruktur dan semi terstruktur. SPK ini mendayagunakan sumber daya individu-individu dengan kemampuan komputer untuk meningkatkan kualitas keputusan.

* 1. **Metode *Brown Gibson***

*Brown Gibson* adalah metode yang digunakan untuk menganalisis alternatif-alternatif lokasi yang dikembangkan berdasarkan konsep “*PreferencesOf Measurement”,* yang dikombinasikan dengan faktor-faktor objektif dan faktor-faktor subjektif. Faktor objektif berupa efektivitas biaya yaitu jumlah total biaya yang dikeluarkan untuk satu alternatif lokasi. Faktor subjektif berupa pembobotan pada pengambil keputusan terhadap kriteria-kriteria yang disyaratkan dalam penentuan lokasi baru.

Metode *Brown Gibson* dikembangkan oleh dua orang peneliti yang bernama Philip Brown dan David Gibson pada tahun 1972. Dasar penerapan metode ini awalnya digunakan untuk menganalisis alternatif lokasi yang dikembangkan berdasarkan “*Preferences Of Measurement”,* yaitu dengan cara dikombinasikan faktor objektif dan faktor subjektif. Metode *Brown Gibson* biasa digunakan untuk membantu analisis data dalam proses pengambilan keputusan yang memiliki multi atribut Proses penilaian kandidat lokasi dengan menggunakan metode *Brown Gibson* akan menggunakan sistem bobot, dimana pada akhir penilaian kandidat lokasi yang memperoleh penilaian terbaik akan menjadi pilihan alternatif terbaik. Dalam menentukan kriteria-kriteria yang akan digunakan untuk menilai kandidat dari alternatif, akan digunakan faktor-faktor kriteria objektif dan faktor-faktor kriteria subjektif (Susilo, 2005).

* 1. **Langkah Kerja Metode *Brown Gibson***

Langkah-langkah untuk mengaplikasikan metode *Brown Gibson* dapat diuraikan sebagai berikut (Susilo, 2005):

1. Mengeliminasi setiap alternatif pilihan yang secara sepintas jelas tidak layak dan layak untuk dipilih, atas dasar pertimbangan-pertimbangan teknis, atau keperluan lainnya dalam kapasitas alternatif yang dibutuhkan, dan bisa dijadikan alasan utama untuk mengeliminasi suatu alternatif dalam daftar nominasi alternatif.
2. Menghitung dan menetapkan *performance measurement* dari faktor objektif (*OFi*) untuk setiap alternatif. Ukuran *performance* untuk faktor objektif dihitung berdasarkan estimasi seluruh perkiraan total biaya-biaya yang dikeluarkan untuk pemilihan alternatif yang dipertimbangkan.

(2.1)

Dimana : Ʃ*OFi* = 1

*Ci* = total estimasi perkiraan biaya

*OFi*  = faktor objektif

*i*  = banyaknya lokasi

1. Menentukan faktor-faktor yang memberi pengaruh signifikan dan harus dipertimbangkan pada saat pemilihan alternatif. Faktor-faktor ini lebih bersifat subjektif. Estimasi dari ukuran faktor subjektif (*SFi*) untuk setiap alternatif pilihan ditentukan dengan menggunakan rumus:

(2.2)

Dimana : Ʃ*SFi* = 1

*i* = jumlah alternatif

*j*  = jumlah faktor subjektif = 1,2,3,…n

*Wj*  =*rating*faktor dengan menggunakan *“forced choice pairwise comparison”*

*Rij*  = rangking faktor subjektif masing-masing alternatif (0 ≤ *Rij* ≤ 1) dan Ʃ*Rij* = 1)

Cara *“forched choice pairwise comparison”* pada prinsipnya adalah membandingkan dan menilai suatu faktor subjektif terhadap faktor subjektif yang secara berpasangan (*pairwise*) yang didasarkan pada:

a. Lebih baik, diberi point = 1

b. Sama baik, diberi point masing-masing = 1

c. Sama jelek, diberi point masing-masing = 0

d. Lebih jelek, diberi point = 0

1. Membuat pembobotan, mana yang lebih baik dipertimbangkan, antara faktor objektif (bobot = *k*) dengan faktor subjektif (bobot = 1 – *k*) dari nilai batas (0<*k*<1). Kombinasikan faktor objektif (*OFi*) dengan faktor subjektif (*SFi*) yang akan menghasilkan “*location preference measure”* (*LPMi*) untuk setiap alternatif yang ada. Secara matematis ditunjukan dengan rumus:

(2.1)

Dimana : Ʃ *LPMi* = 1

*LPMi* =nilai *location preference measure* pada objek alternatif perhitungan

*k* = bobot faktor objektif

*1-k* = bobot faktor subjektif

*OFi* = faktor objektif

*SFi* = faktor subjektif

Keputusan diambil berdasarkan alternatif pilihan yang memiliki nilai *LPMi* terbesar.

* 1. **Metode Pengembangan Perangkat Lunak**

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 ke-2 sudah dilakukan, seperti Gambar 2.1.

Analisis Kebutuhan

Desain Sistem

Penulisan Kode Program

Pengujian Program

Penerapan Program dan Pemeliharaan

**Gambar 2.1 Tahap-Tahap Metode *Waterfall***

1.  Analisis Kebutuhan

Langkah ini merupakan analisis terhadap kebutuhan sistem. Pengumpulan data dalam tahap ini bisa dengan melakukan sebuah penelitian, wawancara atau studi literatur. Sistem analis akan menggali informasi sebanyak-banyaknya dari *user*sehingga akan tercipta sebuah sistem komputer yang bisa melakukan tugas-tugas 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 inilah yang akan menjadi acuan sistem analisis untuk menerjemahkan ke dalam bahasa pemrograman.

2.  Desain Sistem

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

3.  Penulisan Kode Program

Penulisan kode program atau*coding*merupakan penerjemahan *design*dalam bahasa yang bisa dikenali oleh komputer. Dilakukan oleh *programmer* yang akan menterjemahkan transaksi yang diminta oleh *user.* Tahapan inilah 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.

4.  Pengujian Program

Tahapan akhir dimana sistem yang baru akandiuji kemampuan dan keefektifannya sehingga didapatkan kekurangan dan kelemahan sistem yang kemudian dilakukan pengkajian ulang dan perbaikan.

5.  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.

* 1. **Perancangan Sistem**
     1. ***Unified Modeling Language* (UML)**

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

UML adalah salah satu *tools*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 berbagai bahasa pemograman, seperti JAVA, yang dihubungkan secara langsung ke dalam sebuah *object-oriented database*. 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 3 diagram, yaitu;

1. *Use Case Diagram*
2. *Sequence Diagram*
3. *Activity Diagram*
4. *Class Diagram*

UML yang akan digunakan yaitu *Use Case Diagram*, *Sequence Diagram*, *Activity Diagram, Class Diagram*.

1. *Use Case Diagram*

*Use case diagram* menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem dan bukan “bagaimana”.*Use Case Diagram* menggambarkan kebutuhan sistem dari sudut pandang *user*, difokuskan pada proses komputerisasi (*automated processes*) yang menggambarkan hubungan antara *use case* dan *actor Use case* menggambarkan proses sistem (kebutuhan sistem dari sudut pandang *user*) (Nugroho, 2009).

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

|  |  |
| --- | --- |
| **SIMBOL** | **KETERANGAN** |
|  | Aktor, mewakili peran sistem yang lain atau alat ketika berkomunikasi dengan *use case* |
|  | *Use case* abstraksi dari interaksi antara sistem dan actor |
|  | *Association* adalah abstraksi dari penghubung antara aktor dan *use case* |
| <<include>> | Menunjukan bahwa suatu *use case* seluruhnya merupakan fungsionalitas dari *use case* lainnya |
| <<extend>> | Menunjukan bahwa suatu *use case* merupakan tambahan fungsionalitas dari *use case* lainnya jika suatu kondisi terpenuhi |
|  | Menunjukan spesialisasi aktor untuk dapat berpartisipasi dalam *use case* |

1. *Sequence Diagram*

*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 antara dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respon 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/metode dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message* (Nugroho, 2009).

**Tabel 2.2 Simbol *Sequence Diagram***

|  |  |  |
| --- | --- | --- |
| **SIMBOL** | **NAMA** | **KETERANGAN** |
|  | *LifeLine* | Objek *entity*, antarmuka yang saling berinteraksi. |
|  | *Message* | Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktivitas yang terjadi. |
|  | *Message* | Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktivitas yang terjadi. |

1. *Activity Diagram*

*Activity diagram* memodelkan *workflow* proses 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* (Nugroho, 2009).

**Tabel 2.3 Simbol *Activity Diagram***

|  |  |
| --- | --- |
| **SIMBOL** | **KETERANGAN** |
|  | Titik awal |
|  | Titik akhir |
|  | *Activity* Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain |
|  | Pilihan untuk mengambil keputusan dan mengakhiri kondisi |
|  | *Fork* &*Join*; untuk menunjukan kegiatan yang dilakukan secara paralel dan digabungkan kembali |

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 analisis sistem, *designer*, dan *programmer*. Sedangkan pada saat ini, merupakan bagian dari *software trade* (bisnis perangkat lunak). UML memberikan jalur komunikasi dari sistem analisis kemudian *designer*, lalu *programmer* mengenai rancangan *software* yang akan dikerjakan.

Model adalah gambaran abstrak dari suatu dasar masalah dan dunia nyata atau tempat dimana masalah itu timbul bisa disebut dengan domain. Model mengandung objek-objek yang beraktivitas dengan saling mengirimkan *messages* (pesan-pesan). Objek mempunyai sesuatu yang diketahui (atribut) dan sesuatu yang dilakukan (*behaviors* atau *operations*). Atribut hanya berlaku dalam ruang lingkup objek itu sendiri (*state*). Lalu “*blueprint*” dari suatu objek adalah *Class* (kelas). Objek merupakan bagian-bagian dari kelas.

1. *Class Diagram*

*Class diagram* digunakan untuk menampilkan kelas-kelas dan paket-paket di dalam sistem. *Class diagram* memberikan gambaran sistem secara statis dan relasi antara diagram. Biasanya, dibuat beberapa *class diagram* untuk sistem tunggal. Beberapa diagram akan menampilkan subset dari kelas-kelas dan relasinya. Dapat dibuat beberapa diagram sesuai dengan yang diinginkan untuk mendapatkan gambaran lengkap terhadap sistem yang dibangun.

*Class diagram* adalah alat perancangan terbaik untuk tim pengembang. Diagram tersebut membantu pengembang mendapatkan struktur sistem sebelum kode ditulis, dan membantu untuk memastikan bahwa sistem adalah desain terbaik (Nugroho, 2009).

* + 1. **Basis Data**

Basis data adalah kumpulan data yang saling berhubungan yang disimpan secara bersama sedemikian rupa, diberi nama dan tanpa pengulangan yang tidak perlu (redudansi) untuk memenuhi berbagai kebutuhan, dengan tujuan utamanya adalah memelihara informasi dan membuat informasi tersebut tersedia saat dibutuhkan.

Basis data terdiri dari beberapa unsur yaitu ( Sutanta, 1996):

1. *Enterprise*

*Enterprise* (badan, lembaga) diartikan sebagai sembarang badan organisasi seperti instansi pemerintah, sekolah, bank, rumah sakit, industri atau badan usaha lain yang merupakan unit kerja yang dalam batas-batas tertentu bekerja sebagai satu kesatuan.

1. *Entity*

*Entity* (kesatuan) merupakan elemen atau bagian dalam *enterprise* yang kesemuanya direkam. *Entity-entity* dalam suatu *interprise* saling terkait antara yang satu dengan yang lain melalui satu atau lebih relasi.

1. *Attribute*

*Attribute* adalah unsur-unsur *entity* yang menjelaskan ciri atau watak yang dipilih untuk direkam. Attribut ini merupakan satuan terkecil dari pada data yang mempunyai arti bagi pengguna data.

1. Nilai Data

Nilai data adalah sekumpulan dari karakter-karakter atau nilai yang diisikan pada suatu elemen data.

1. *Record Data*

*Record data* merupakan sekumpulan harga data yang saling berhubungan dalam satu *entity*.

* 1. **MySQL**

MySQL adalah sebuah implementasi dari *relational database management system* (RDBMS) yang didistribusikan secara gratis. Setiap pengguna dapat secara bebas menggunakan MySQL, namun dengan batasan perangkat lunak tersebut tidak boleh dijadikan produk turunan yang bersifat komersial. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam basisdata yang telah ada sebelumnya; SQL (*Structured Query Language*). SQL adalah sebuah konsep pengoperasian basis data, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis. Keistimewaan MySQL antara lain yaitu (Saputra 2012):

1. Portabilitas. MySQL dapat berjalan stabil pada berbagai sistem operasi seperti Windows, Linux, FreeBSD, Mac Os X Server, Solaris, Amiga, dan masih banyak lagi.
2. Perangkat lunak sumber terbuka (*open source*). MySQL didistribusikan sebagai *open source* sehingga dapat digunakan secara gratis.
3. *Multi-user.* MySQL dapat digunakan oleh beberapa pengguna dalam waktu yang bersamaan tanpa mengalami masalah atau konflik.
4. *Performance tuning*, MySQL memiliki kecepatan yang menakjubkan dalam menangani *query* sederhana, dengan kata lain dapat memproses lebih banyak SQL per satuan waktu.
5. Ragam tipe data. MySQL memiliki ragam tipe data yang sangat kaya, seperti *signed / unsigned integer, float, double, char, text, date, timestamp,* dan lain-lain.
6. Perintah dan fungsi. MySQL memiliki operator dan fungsi secara penuh yang mendukung perintah *Select* dan *Where* dalam perintah (*query*).
7. Keamanan. MySQL memiliki beberapa lapisan keamanan seperti *password* yang terenkripsi.
8. Skalabilitas dan pembatasan. MySQL mampu menangani basis data dalam skala besar, dengan jumlah *record* lebih dari 50 juta dan 60 ribu tabel serta 5 milyar baris. Selain itu batas indeks yang dapat ditampung mencapai 32 indeks pada tiap tabelnya.
9. Konektivitas. MySQL dapat melakukan koneksi dengan klien menggunakan protokol TCP/IP, *Unix socket* (UNIX).
10. Lokalisasi. MySQL dapat mendeteksi pesan kesalahan pada klien dengan menggunakan lebih dari dua puluh bahasa. Meskipun demikian, bahasa Indonesia belum termasuk di dalamnya.
11. Antar Muka MySQL memiliki antar muka (*interface*) dan bahasa pemrograman dengan menggunakan fungsi API (*Application Programming Interface*).
    1. ***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 berbasis *desktop* 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 *programmer* untuk menulis, meng-*compile*, mencari kesalahan dan menyebarkan program *Netbeans* yang ditulis dalam bahasa pemrograman Java, serta mendukung bahasa pemrograman lainnya dan program ini pun bebas untuk digunakan dan untuk membuat profesional *desktop*, *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 dan terus bertambah. *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 layanan-layanan yang umum bagi aplikasi *desktop* dan mengizinkan pengembang untuk fokus ke logika yang spesifik terhadap aplikasi( Setiowati, 2012).

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 metode.
2. *Bookmarking*: fitur yang digunakan untuk menandai baris yang suatu saat hendak dimodifikasi.
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 dapat meng-*generate constructor*, *setter and getter* metode dan yang lainnya.
5. *Error stripe*: fitur yang akan menandai baris yang error dengan memberi *highlight* merah.
   1. **XAMPP**

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 (Petter, 2007).

XAMPP adalah singkatan yang masing-masing hurufnya adalah:

* + - 1. X: Program ini dapat dijalankan pada beberapa 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, dapat memanfaatkan MySQL untuk menambahkan, mengubah, dan menghapus data yang berada dalam *database*.
      4. P: PHP, bahasa pemrograman web. Bahasa pemrograman PHP merupakan bahasa pemrograman untuk membuat web yang bersifat *server-side scripting*.
      5. P: Perl, bahasa pemrograman.