## BAB II

**TINJAUAN PUSTAKA**

## Tinjauan Pustaka

Penulis ingin melakukan peningkatan kemudahan untuk mengetahui dengan cepat penyakit pada anak, Karena kebiasaan orang tua panik saat anaknya tiba tiba sakit tanpa ada ilmu kenapa anaknya sakit, sehingga perlu ada system untuk mempermudah orang tua mengetahui apa yang terjadi pada anaknya [1] Untuk mengatasi permasalahan tersebut, diusulkan sistem informasi dengan menggunakan sistem kecerdasan buatan. Dimana sistem ini dapat membantu dokter yang belum berpengalaman dalam mendukung diagnosa dokter terhadap jenis penyakit mata glaukoma. Sehingga dokter dapat melakukan diagnosa secara spesifik. [2]

Di Tinjauan yang lain Penulis ingin membuat sebuah system yang dimana diperuntukan bagi rumah sakit yang kekurangan dokter spesialis Kanker darah, karena di Studi kasus penulis yang menjelaskan bagaimana bahayanya kanker darah pada anak jika tak cepat diatasi sedangkan untuk mendiagnosa sendiri perlu dokter ahli sehingga system pakar akan sangat membantu dalam mendiagnosa penyakit ini[3] tapi dalam *BEST FIRST SEARCH* data yang digunakan cukup banyak sehingga diperlukan Kombinasi dengan *BEST FIRST SEARCH* yang menggunakan data yang lebih sedikit [2]

Penulis membangun system ini karena rentannya balita dan bayi dalam terserang penyakit, karena kelemehan fisik inilah penulis membangun system ini untuk memudahkan orang tua dalam mengetahui penyakit apa yang terjadi pada anaknya. Tetapi tak ada pembahasan lebih lanjut tentang penyakit penyakit apa saja yang bisa dikenali [4] maka sebab itu lah diperlukan *memory* yang cukup besar agar system bisa belajar sendiri ketika mengenali kasus yang belum dikenali [5]

**Tabel 2.1 Penelitian Sebelumnya**

|  |  |  |
| --- | --- | --- |
| **NO** | **JUDUL PENELITIAN** | **PERBANDINGAN FITUR** |
| 1 | Sistem Pakar Untuk Mendiagnosa Penyakit Pada Balita Berbasis *DFS* | 1. User digunakan oleh banyak kalangan termasuk orang awam 2. Hanya untuk mendiagnosa penyakit balita umur 2 – 5 tahun |
| 2 | Sistem Pakar Penyakit Mata Glaukoma Dengan Metode *Breathd First Search )* dan *Fuzzy Tsukamoto* | 1. Sistem dibuat dan hanya digunakan oleh pihak rumah sakit 2. Sistem pakar khusus penyakit mata glaucoma |
| 3 | Diagnosis Kanker Darah pada Anak menggunakan Inferensi *Forward Chaining* | 1. Hanya digunakan oleh dokter ahli 2. Hanya mendiagnosis penyakit kanker darah pada anak |

## Konsep Dasar Kecerdasan Tiruan

*Artificial Intelligence* merupakan bagian tengah atau inti dari ilmu komputer dan merupakan salah satu ilmu dasar dari ilmu komputer yang harus dipahami, bila kita ingin membuat perangkat lunak dengan penerapan konsep *Artificial Intelligence* dalam memecahkan persoalan yang ada di dunia ini, sehingga keberadaannya mutlak harus ada. Karena kecerdasan tiruan adalah ilmu yang berdasarkan proses manusia berpikir, maka penelitian bagaimana proses manusia berpikir adalah hal yang pokok.

Semua proses berpikir menolong manusia untuk menyelesaikan sesuatu masalah. Pada saat otak manusia mendapat informasi dari luar, maka suatu proses berpikir memberikan petunjuk tindakan atau respon apa yang dilakukan. Hal ini merupakan suatu reaksi otomatis dan respon yang spesifik dicari untuk menyelesaikan masalah tertentu. Hasil akbar dari semua proses berpikir tersebut disebut tujuan (goal). Berdasarkan fakta-fakta yang ada keterangan diatas, maka komputer dapat digunakan sebagai alat bantu dalam mengambil keputusan.

## 2.3 Sistem pakar

### 2.3.1. Definisi sistem pakar

Sistem pakar atau sistem berbasis pengetahuan adalah yang paling banyak aplikasinya dalam membantu menyelesaikan masalah-masalah dalam dunia nyata. Perangkat lunak ini dapat sekali dijalankan oleh perangkat komputer pribadi, sehingga untuk aplikasi kecerdasan tiruan ini dapat dilakukan dengan mudah dan dengan biaya yang relatif lebih murah. Sistem pakar adalah sebuah program yang dapat :

1) Menangani masalah dunia nyata, masalah yang komplek yang sangat membutuhkan interpretasi pakar.

2) Menyelesaikan masalah dengan menggunakan komputer dengan model penalaran manusia dan mencapai kesimpulan yang sama dengan yang dicapai oleh seorang pakar jika berhadapan langsung dengan sebuah masalah yang sedang di hadapinya.

Komputer yang berbasis sistem pakar adalah program komputer yang mempunyai pengetahuan yang berasal dari manusia yang berpengetahuan luas dalam suatu domain tertentu, dimana pengetahuan di sini adalah pengetahuan manusia yang sangat minim penyebarannya, mahal dan serta susah untuk di dapatkannya.

Kondisi-kondisi dimana sistem pakar dapat membantu menusia dalam menyelesaikan masalahnya, antara lain:

1) Kebutuhan tenaga ahli yang banyak, tetapi pakar yang tesedia jumlahnya hanya sedikit dan sangat terbatas.

2) Pemakaian pakar yang berlebihan dalam membuat keputusan, walaupun dalam suatu tugas yang rutin.

3) Pertimbangan kritis harus dilakukan dalam waktu yang singkat untuk menghindari hal-hal yang tidak diinginkan.

4) Hasil yang optimal, seperti dalam hal perencanaan atau konfigurasi.

5) Sejumlah besar data yang harus diteliti oleh pakar secara berkelanjutan.

### 2.3.2 Pembentukan umum expert system

Dengan metode *Forward-Channing*, data-driven karena inference engine menggunakan informasi yang ditentukan oleh user untuk memindahkan keseluruhan jaringan dari logika *’AND’* dan *’OR’* sampai sebuah terminal di tentukan sebagai obyek.

Bila inference engine tidak dapat menentukan obyek maka akan meminta informasi lain. Aturan (*Rule*) di mana menentukan obyek, membentuk lintasan (*Path*) yang mengarah ke obyek. Oleh karna itu, hanya satu cara untuk mencapai obyek adalah memenuhi semua aturan.

### 2.3.3 Ciri-ciri sistem pakar

Ciri-ciri dari sistem pakar adalah:

1) Terbatasnya pada domain keahlian tertentu.

2) Dapat memberikan penalaran data yang tidak pasti.

3) Dapat mengemukakan rangkaian alasan-alasan yang tidak diberikannya dengan cara yang tidak dapat di pahami.

4) Berdasarkan kaidah-kaidah/ketentuan/rule yang berlaku.

5) Dirancang untuk dapat dikembangkan secara bertahap.

6) Pengetahuan dan mekanisme penalaran jelas terpisah.

7) Keluaran bersifat anjuran.

8) Sistem dapat mengaktifkan kaidah secara searah yang sesuai di tuntun oleh dialog dengan user.

### 2.3.4 Klasifikasi sistem pakar

1) Sistem Pakar Diagnosis

Digunakan untuk melakukan pencarian penyelesaian masalah atau kerusakan suatu alat atau mesin tertentu. Prinsipnya adalah menemukan masalah atau kerusakan apa yang terjadi. Sistem ini merupakan jenis sistem pakar yang paling popular saat ini, melakukan diagnosis, menggunakan deskripsi keadaan karakteristik tingkah laku, atau pengetahuan tentang pembuatan komponen sehingga dapat menentukan kemungkinan kerusakan pada sistem.

Contoh: Diagnosis medis kedokteran, diagnosis kerusakan alat elektronik dan perancangan sistem komunikasi dan radio.

2) Sistem Pakar Pengajaran

Digunakan untuk mengajar, mulai dari murid Sekolah Dasar sampai mahasiswa Perguruan Tinggi. Kelebihan dari sistem pakar yang digunakan untuk mengajar adalah membuat diagnosa penyebab kekurangan dari seorang siswa, kemudian memberi cara untuk memecahkannya.

Contoh: Contoh sistem pakar untuk pengajaran bahasa.

3) Sistem Pakar Interpretasi

Digunakan untuk menganalisa data yang tidak lengkap, tidak teratur dan data yang kontradiktif, misalnya untuk interpretasi pengawasan, pengertian bahasa, analisis citra, dan lain-lain.

Contoh: sistem pakar interpretasi lingkungan pengendapan delta.

4) Sistem Pakar Prediksi

Digunakan untuk melakukan peramalan suatu keadaan dengan ditunjang oleh data yang diperoleh sebelumnya, sistem ini dapat memberikan kemungkinan penyelesaian tertentu.

Contoh: Prediksi Penyebab dan Solusi Ketidaknyamanan Kerja dengan Aplikasi Sistem Pakar.

5) Sistem Pakar Perencanaan

Merupakan suatu sistem yang sangat luas mulai dari perencanaan mesin-mesin sampai manajemen bisnis. Penggunaan sistem pakar jenis ini menghemat biaya, waktu, dan material, sebab pembuatan model sudah tidak diperlukan lagi.

Contoh: Perencanaan Sistem Distribusi Menggunakan Sistem Pakar.

6) Sistem Pakar Kontrol

Digunakan untuk mengendalikan kegiatan yang membutuhkan presisi waktu yang tinggi. Sistem ini memperhatikan tingkah laku sistem yang dapat dapat disebut normal atau tidak normal. Sistem ini bergantung pada waktu untuk menginterpretasikan tingkah laku yang diamati. Misalnya pada pengendali manajemen bisnis dan pengendali pada industri berteknologi tinggi.

### Keuntungan Sistem Pakar

Secara garis besar, banyak manfaat yang dapat di ambil dengan adanya sistem pakar, antara lain:

1) Memungkinkan orang awam bisa mengerjakan pekerjaan para ahli.

2) Bisa melakukan proses secara berulang-ulang secara otomatis.

3) Menyiapkan pengetahuan dan keahlian para pakar.

4) Meningkatkan output dan produktifitas.

5) Meningkatkan kualitas.

6) Mampu mengambil dan melestarikan keahlian para pakar (terutama yang termasuk keahlian langka).

7) Mampu beroperasi dalam lingkungan yang berbahaya.

8) Memiliki kemampuan untuk mengakses pengetahuan

9) Memiliki reliabilitas.

10) Meningkatkan kapabilitas sistem komputer.

11) Memiliki kemampuan untuk bekerja dengan informasi yang tidak lengkap dengan mengandung ketidakpastian.

12) Sebagai media pelengkap dalam pelatihan.

13) Meningkatkan kapabilitas dalam penyelesaian masalah.

14) Menghemat waktu dalam pengambilan suatu keputusan.

**Tabel 2.2 Perbedaan Sistem Konvensional dan Sistem Pakar**

|  |  |
| --- | --- |
| Sistem Konvensional | Sistem Pakar |
| 1. Informasi dan pemrosesan  umumnya digabung dalam suatu  program yang suquential  2. Program tidak perna salah  (kecuali karena programernya)  3. Tidak menjelaskan kenapa input  dibutuhkan atau bagaimana  hasil diperoleh  4. Membutuhkan semua input data  5. Perubahan pada program sangat  merepotkan  6. Sistem bekerja jika sudah  lengkap  7. Eksekusi secara algoritma (step  by step)  8. Manipulasi efektif pad data  base yang besar  9. Efesiensi adalah tujuan utama  10. Data kuantitatif  11. Representasi data dalam  numerik  12. menangkap, menambah dan  mendistribusikan data numerik  atau informasi | 1. Knowledge base terpisah dari  mekanisme pemrosesan  2. Program bisa saja melakukan  kesalahan  3. Penjelasan (Explanation)  merupakan bagian dari Expert  system  4. Tidak harus membutuhkan data  input atau data fakta  5. Perubahan pada rule dapat  dilakukan dengan mudah  6. Sistem dapat bekerja pada rule  yang sedikit  7. Eksekusi dilakukan secara  heuristik dan logik  8. Manipulasi efektif pada  knowledge-base yang besar  9. Efektifitas adalah tujuan utama  10. Data kualitatif  11. Representasi pengetahuan  dalam simbol  12. Menangkap, menambah, dan  mendistribusi pertimbagan  (judgment) dan pengetahuan |

## 2.4 Best First Search

*Best first search* merupakan kombinasi dari metode *Depth First Search* dan *Breadth First Search* dimana pencarian diperbolehkan mengunjungi node pada level lebih rendah jika node pada level lebih tinggi memiliki nilai heuristik lebih buruk. Algoritma pada *best first search* yaitu :

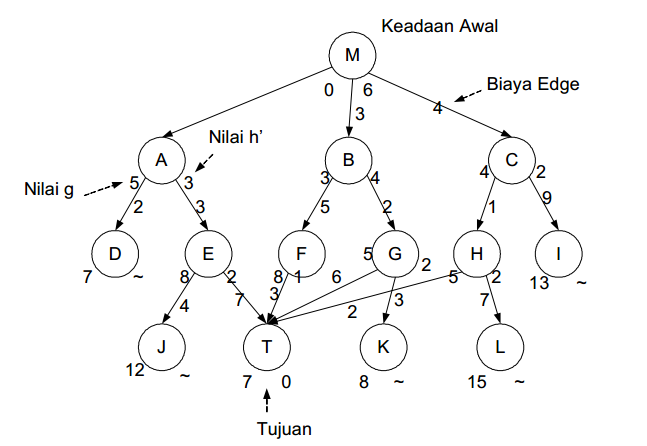
1. Buat sebuah stack kemudian inisialisasikan node akar sebagai node pertama.

2. Bila node pertama ≠ GOAL, node diganti dan dihapus dengan anak- anaknya.

3. Kemudian, keseluruhan node pada stack diurutkan secara ascending berdasarkan fungsi heuristik yang digunakan.

4. Bila node pertama ≠ GOAL, ulangi langkah 2. Bila node pertama = GOAL, cari solusi dengan menelusuri jalur dari GOAL ke node akar.

Teknik ini akan mengunjungi dimulai dari node terdalam selama node tersebut merupakan node yang terbaik. Jika node yang sedang dikunjungi ternyata tidak mengarah kepada solusi yang diinginkan, maka akan dilakukan runut balik ke arah node akar untuk mencari node anak lainnya yang lebih menjanjikan daripada node yang terakhir dikunjungi. Bila tidak ada juga, maka akan terus mengulang mencari kearah node akar sampai ditemukan node, yang lebih baik untuk dibangkitkan suksesornya. Teknik ini bertolak belakang dengan teknik *Depth First Search* yang mencari sampai kedalaman yang, terdalam sampai tidak ada node suksesor yang bisa dibangkitkan sebelum melakukan runut balik, dan *Breadth First Search* yang tidak akan melakukan pencarian secara mendalam sebelum pencarian secara melebar selesai. Dalam pengembangan Sistem Pakai ini, penulis akan menggunakan teknik *Best First Search.*

**

Gambar 2.1 Contoh Alur *Best First Search*

## 2.5 Platform Android

*Android* merupakan salah satu sistem operasi yang terkenal dikalangan perangkat mobile yang merupakan pesaing dari sistem operasi perangkat *mobile* lainnya seperti *Windows Phone, iOS, BlackBerry, MeeGo*, Bada dan Symbian. Namun berbeda dengan sistem operasi mobile lainnya, karena *Android* bersifat *Open Source* yang memungkinkan untuk dikembangkan lebih lanjut oleh pihak ketiga.

*Android* adalah sebuah sistem operasi untuk perangkat *mobile* berbasis linux yang mencakup sistem operasi, *middleware* dan aplikasi. *Android* diakusisi oleh Google pada Juli 2005, dan baru dirilis perdana pada 5 November 2007. Android berlisensi di bawah GNU, General Public Lisensi Versi 2 (GPLv2), yang memperbolehkan pihak ketiga untuk mengembangkannya dengan menyertakan term yang sama. Pendistribusiannya di bawah Lisensi Apache Software (ASL/Apache2), yang memungkinkan untuk

distribusi kedua dan seterusnya. Android dirancang dengan arsitektur sebagai berikut

1. Application dan Widgets, merupakan layer dimana kita berhubungan dengan aplikasi saja, seperti aplikasi untuk browsing. Selain itu, fungsi- fungsi seperti telepon dan sms juga terdapat pada layer ini.

2. Application Frameworks, merupakan layer dimana para pembuat aplikasi melakukan pengembangan/ pembuatan aplikasi yang akan dijalankan di sistem operasi Android. Beberapa komponen yang terdapat pada layer ini adalah, Views, Content Provider, Resource Manager, Notification Manager dan Activity Manager.

3. *Libraries*, merupakan *layer* dimana fitur-fitur Android berada yang dapat digunakan untuk menjalankan aplikasi. Library yang disertakan seperti library untuk pemutaran audio dan video, tampilan, grafik, SQLite, SSL dan Webkit, dan 3D.

4. *Android Run Time*, merupakan layer yang berisi *Core* *Libraries* dan *Dalvik* *Virtual Machine* (DVK). Core libraries berfungsi untuk menerjemahkan bahasa *Java/C*. Sedangkan DVK merupakan sebuah virtual mesin berbasis register yang dioptimalkan untuk menjalankan fungsi-fungsi secara efisien.

5. *Linux Kernel*, merupakan layer yang berfungsi sebagai abstraction/ pemisah antara hardware dan software. *Linux kernel* inilah yang merupakan inti sistem operasi dari Android yang berfungsi untuk

mengatur sistem proses, memory, resouce, dan driver. Linux kernel yang digunakan Android adalah linux kernel release 2.6.

fitur versi sebelumnya yaitu *Honeycomb* untuk smartphone dan menambahkan fitur baru termasuk membuka kunci dengan pengenalan wajah, jaringan data pemantauan penggunaan dan kontrol, terpadu kontak jaringan sosial, perangkat tambahan fotografi, mencari *email* secara *offline*, dan berbagi informasi dengan NFC .

Beberapa keunggulan Platform Android adalah sebagai berikut (Safaat, 2001:3):

1. Lengkap (*Complete Platform*). Para desainer dapat melakukan pendekatan

yang komprehensif ketika sedang mengembangkan platform Android. Android menyediakan banyak tools dalam membangun software dan merupakan sistem operasi yang aman.

1. Terbuka (*Open Source Platform*). Platform *Android* disediakan melalui lisensi open source.
2. Bebas (*Free Platform*). Android merupakan platform atau aplikasi yang bebas untuk dikembangkan. Tidak ada lisensi atau biaya royalti untuk dikembangkan pada platform Android.

## 2.6 Pengujian Software

Pengujian software sangat diperlukan untuk memastikan software/aplikasi yang sudah/sedang dibuat dapat berjalan sesuai dengan fungsionalitas yang diharapkan. Pengembang atau penguji software harus menyiapkan sesi khusus untuk menguji program yang sudah dibuat agar kesalahan ataupun kekurangan dapat dideteksi sejak awal dan dikoreksi secepatnya. Pengujian atau testing sendiri merupakan elemen kritis dari jaminan kualitas perangkat lunak dan merupakan bagian yang tidak terpisah dari siklus hidup pengembangan software seperti halnya analisis, desain, dan pengkodean [7].

### 2.6.1 White Box Testing

White box Ttsting adalah salah satu cara untuk menguji suatu aplikasi atau software dengan cara melihat modul untuk dapat meneliti dan menganalisa kode dari program yang di buat ada yang salah atau tidak. Apabila modul yang telah dan sudah di hasilkan berupa output yang tidak sesuai dengan yang di harapkan maka akan dikompilasi ulang dan dilakukan pengecekan kembali kode-kode tersebut hingga sesuai dengan yang diharapkan.[7]

### 2.6.2 Black Box Testing

Black box testing berfokus pada spesifikasi fungsional dari perangkat lunak. Tester dapat mendefinisikan kumpulan kondisi input dan melakukan pengetesan pada spesifikasi fungsional program.

Black box testing bukanlah solusi alternatif dari white box testing tapi lebih merupakan pelengkap untuk menguji hal-hal yang tidak dicakup oleh white box testing. [7]

Black box testing cenderung untuk menemukan hal-hal berikut [7] :

1. Fungsi yang tidak benar atau tidak ada.

2. Kesalahan antar muka (interface errors).

3. Kesalahan pada struktur data dan akses basis data.

4. Kesalahan performansi (performance errors).

5. Kesalahan inisialisasi dan terminasi.

## 2.7 SQlite

*SQLite* merupakan sebuah sistem manajemen basisdata relasional yang bersifat *ACID-compliant* dan memiliki ukuran pustaka kode yang relatif kecil, ditulis dalam bahasa *C. SQLite* merupakan proyek yang bersifat public domain yang dikerjakan oleh D. Richard Hipp.

Tidak seperti pada paradigma *client-server* umumnya, Inti SQLite bukanlah sebuah sistem yang mandiri yang berkomunikasi dengan sebuah program, melainkan sebagai bagian integral dari sebuah program secara keseluruhan. Sehingga protokol komunikasi utama yang digunakan adalah melalui pemanggilan *API* secara langsung melalui bahasa pemrograman. Mekanisme seperti ini tentunya membawa keuntungan karena dapat mereduksi overhead, latency times, dan secara keseluruhan lebih sederhana. Seluruh elemen basisdata (definisi data, tabel, indeks, dan data) disimpan sebagai sebuah file. Kesederhanaan dari sisi disain tersebut bisa diraih dengan cara mengunci keseluruhan file basis data pada saat sebuah transaksi dimulai.

## *2.8 Entity Relationship Diagram* (ERD)

*Entity Relationship Diagram* (ERD) adalah suatu model jaringan yang disimpan dalam sistem secara abstrak dan ERD merupakan model jaringan data yang menekankan pada struktur-struktur dan *relationship* data.

Biasanya ERD ini digunakan oleh professional sistem untuk berkomunikasi dengan pemakaian eksekutif tingkat tinggi dalam suatu organisasi. ERD juga menguntungkan bagi profesional sistem karena ERD memperlihatkan hubungan antar data store pada DFD. Ada 3 (tiga) macam simbol yang digunakan dalam ERD yaitu[7] :

1. Entitas (*entity*)

Entitas merupakan individu yang mewakili sesuatu yang nyata eksistensinya dan dapat dibedakan dari sesuatu yang lain. Entitas dapat berupa objek, orang, konsep, abstrak atau kejadian.

1. Relasi (*relasionship*)

Adalah hubungan atau asosiasi suatu entitas dengan dirinya sendiri atau dengan entitas lainnya. *Relationship* digambarkan sebagai garis yang menghubungkan entitas-entitas yang dipandang memiliki hubungan antara satu dengan yang lainnya.

1. Atribut (*attribute*)

Atribut mendeskripsikan karakteristik dari suatu entitas. Umumnya penetapan atribut bagi sebuah entitas didasarkan pada fakta yang ada.

Menurut Ladjamudin (2005:142), ada 3 (tiga) jenis relasi hubungan atribut dalam satu file, yaitu relasi satu ke satu, relasi satu ke banyak dan relasi banyak ke banyak [7].

a. Relasi Satu ke Satu

Hubungan antara *file* pertama dengan kedua adalah satu banding satu dengan relasi antar keduanya diwakilkan dengan tanda panah tunggal.

b. Relasi Satu ke Banyak

Hubungan antara *file* pertama dengan *file* kedua adalah satu banding banyak atau dapat pula dibalik, banyak banding satu dengan relasi antara keduanya diwakilkan dengan tanda panah ganda untuk menunjukkan hubungan banyak tersebut.

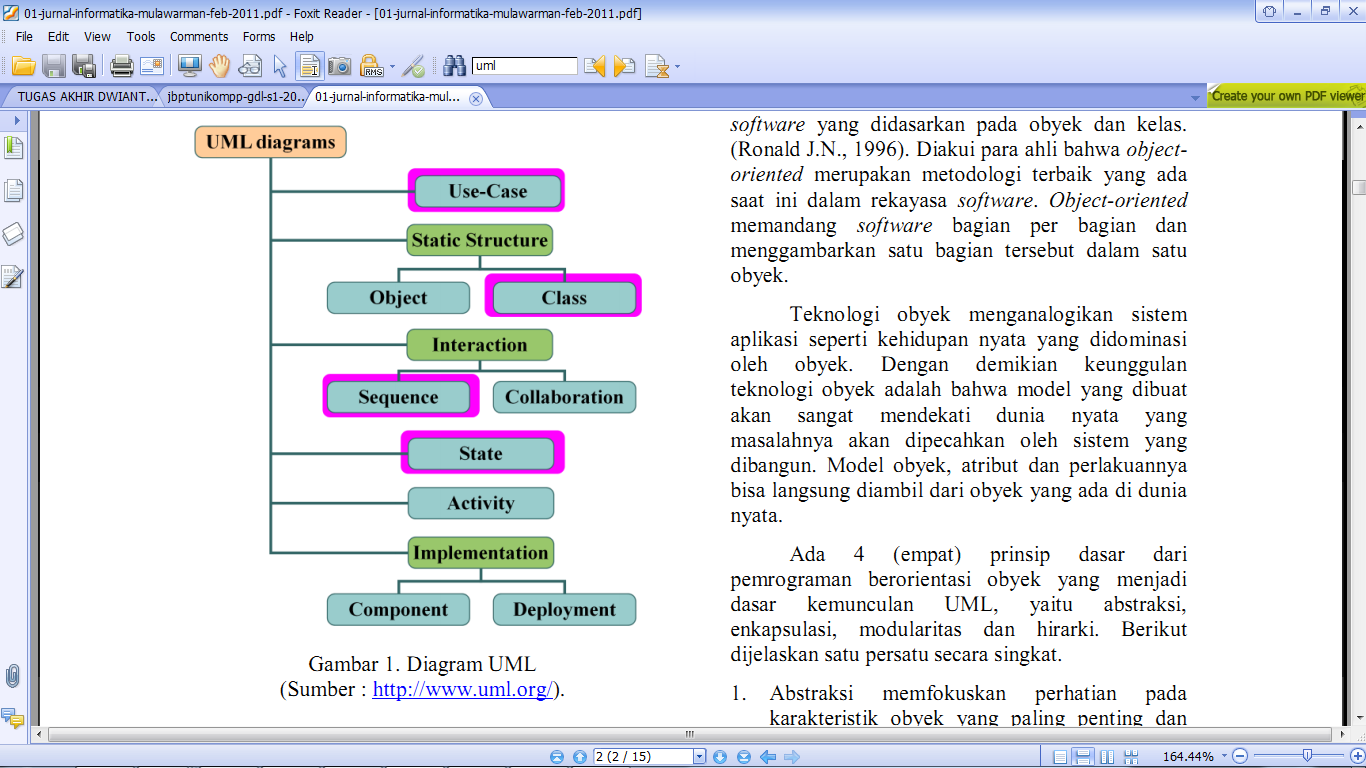
c. Relasi Banyak ke Banyak

Hubungan antara *file* pertama dengan *file* kedua adalah banyak banding banyak dengan relasi antar keduanya diwakilkan dengan tanda panah ganda untuk menunjukkan hubungan banyak tersebut.

Penentuan struktur *file* dilakukan dengan menentukan *file-file* apa saja yang harus disediakan dan yang akan digunakan dalam aktivitas kerja jika sistem diimplementasikan. Pemilihan *file* dapat ditentukan dari skema ER yang telah dalam bentuk tabel normal. Dari tabel tersebut baru akan dibuatkan struktur filenya. Struktur *file* berisi data yang harus disimpan oleh perusahaan serta cara data tersebut akan diakses.

### 2.8.1 Unified Modeling Language (UML)

Menurut Haviluddin (2011:1) *Unified Modeling Language* (UML) adalah suatu alat untuk memvisualisasikan dan mendokumentasikan hasil analisa dan desain yang berisi sintak dalam memodelkan sistem secara visual juga merupakan satu umpulan konvensi pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem software yang terkait dengan objek [7]. Berikut gambar dari diagram UML.



Gambar 2.2.Gambar Diagram UML [7]

Tujuan dari penggunaan diagram menurut Schmuller J.(2004), “*The purpose of the diagram is to present multiple views of a system; this set of multiple views is called a model”* [7] yang memiliki arti dalam bahasa Indonesia “Tujuan dari diagram adalah untuk menampilkan atau menyajikan beberapa tampilan dari suatu sistem; beberapa tampilan ini disebut dengan model”.

Menurut Sugrue J.(2009) tujuan utama dari desain UML adalah [7] :

1. Menyediakan analisis dan desain sistem suatu bahasa permodelan visual yang ekspresif sehingga pengguna dapat mengembangkan dan melakukan pertukaran model data yang bermakna;
2. Sebagai penyedia mekanisme yang spesialisasi untuk memperluas konsep tersebut;
3. Memberikan dasar formal untuk pemahaman dalam bahasa permodelan;
4. Mendorong pertumbuhan pasar terhadap penggunaan alat desain sistem yang berorientasi objek;
5. Mendukung konsep pembangunan tingkat yang lebih tinggi seperti kolaborasi, kerangka, pola dan komponen terhadap suatu sistem.

### *Use Case* Diagram

*Use case* adalah abstraksi dari interaksi antara sistem dan aktor. *Use case* bekerja dengan cara mendeskripsikan tipe interaksi antara *user* sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. *use case* merupakan konstruksi untuk mendeskripsikan bagaimana sistem akan terlihat di mata *user*. Sedangkan *use case* diagram memfasilitasi komunikasi diantara analis dan pengguna serta antara analis dan klien.

Diagram *use case* berguna dalam tiga hal, antara lain :

1. Menjelaskan fasilitas yang ada (*requirements*)

*Use case* baru selalu menghasilkan fasilitas baru ketika sistem di analisa, dan design menjadilebih jelas.

1. Komunikasi dengan klien

Penggunaan notasi dan simbol dalam diagram *use case* membuat pengembang lebih mudah berkomunikasi dengan klien-kliennya.

1. Membuat test dari kasus-kasus secara umum

Kumpulan dari kejadian-kejadian untuk *use case* bisa dilakukan test kasus layak untuk kejadian-kejadian tersebut.

Menurut Menurut Haviluddin (2011:6) komponen-komponen pembentuk *Use Case* terdiri dari aktor dan usecase [21].

1. *Actor*

*Actor* merupakan penentu peran yang dimainkan oleh *user* atau sistem lain yang berinteraksi dengan subjek. *Actor* merupakan segala sesuatu yang berinterasi langsung dengan sistem aplikasi komputer yang bertugas memberikan informasi kepada sistem serta dapat memberikan perintah kepada sistem untuk melakukan suatu tugas.



Gambar 2.3. *Use case actor*

1. *Use Case*

*Use Case* merupakan deskripsi fungsi dari sebuah sistem yang bekerja dengan cara mendeskripsikan tipikal interaksi antara pengguna sebuah sistem dengan sistem itu sendiri melalui sebuah cerita mengenai bagaimana sistem tersebut digunakan.



Gambar 2.3. *Use case*

### *Sequence* Diagram

Diagram *Class* dan diagram *object* merupakan suatu gambaran model statis. Namun ada jugayang bersifat dinamis, seperti diagram *interaction*. diagram *sequence* merupakan salah satu diagram *interaction* yang menjelaskan bagaimana suatu operasi itu dilakukan. Diagram ini diatur berdasarkan waktu. Objek-objek yang berkaitan dengan proses berjalannya operasi diurutkan dari kiri ke kanan berdasarkan waktu terjadinya dalam pesan yang terurut.

Menurut Nugroho (2009:38), tujuan penggunaan *sequence* diagram adalah sebagai berikut [7] :

a. Mengkomunikasikan *requirement* kepada tim teknis karena diagram ini dapat lebih mudah untuk menjadi model desain serta mudah digunakan dalam melakukan permodelan suatu sistem.

b. Merupakan diagram yang paling cocok untuk mengembangkan model deskripsi *use case* menjadi spesifikasi desain.

c. Analisa dan desain, memfokuskan pada identifikasi method di dalam sebuah sistem. *Sequence* diagram biasanya dipakai untuk memodelkan :

1. Deskripsi tentang sistem yang ada pada sebuah/beberapa *use case* pada *use case* diagram, yang menggambarkan hubungan antara *actor* dan *use case* diagram;
2. Logika dari *method* (*operation*, *function* atau *procedure*);
3. Logika dari *service* (*high level method*).

Komponen-komponen yang digunakan dalam melakukan permodelan menggunakan *sequence* diagram adalah sebagai berikut :

1. *Actor*

*Actor* merupakan segala sesuatu yang berinterasi langsung dengan sistem aplikasi komputer yang bertugas memberikan informasi kepada sistem serta dapat memberikan perintah kepada sistem untuk melakukan suatu tugas.



Gambar 2.4. *Actor* pada *sequence* diagram

1. *Object*

*Object* Merupakan komponen dalam melakukan permodelan menggunakan *sequence* diagram dimana komponen ini digunakan sebagai objek-objek yang akan digunakan didalam pembuatan diagram.



Gambar 2.5. *Object* pada *sequence* diagram

1. *Stimulus*

*Stimulus* Merupakan komponen dalam melakukan permodelan menggunakan *sequence* diagram dimana komponen ini digunakan sebagai penanda adanya hubungan yang terjadi antara dua buah *object* maupun *actor* dimana terjadinya komunikasi berupa pengiriman pesan atau *message*.



Gambar 2.6. *Stimulus*

1. *Self Stimulus*

*Self stimulus* Merupakan komponen dalam melakukan permodelan menggunakan *sequence* diagram dimana komponen ini memiliki fungsi yang sama dengan *stimulus* yaitu digunakan sebagai penanda adanya hubungan yang terjadi antara dua buah *object* maupun *actor*. Namun, komunikasi berupa pengiriman pesan atau *message* dilakukan kepada dirinya sendiri, bukan pada obejek lainnya.



Gambar 2.7. *Self stimulus*

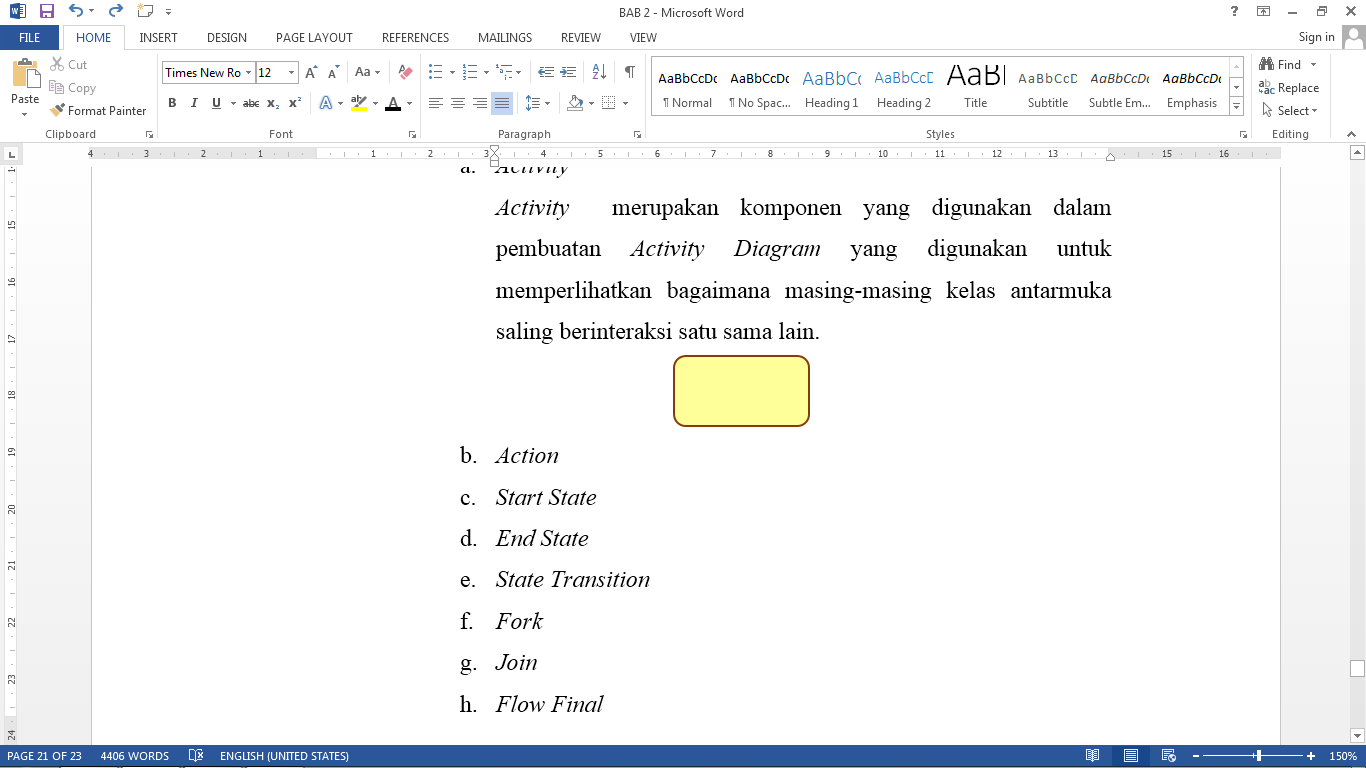
### *Activity* Diagram

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

Komponen-komponen yang digunakan dalam melakukan permodelan menggunakan *activity* diagram adalah sebagai berikut:

1. *Activity*

*Activity* merupakan komponen yang digunakan dalam pembuatan *activity* diagram yang digunakan untuk memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain.



Gambar 2.8. *Activity* pada *activity* diagram

1. *Action*

*Action* merupakan komponen yang berisikan suatu *statement* dari suatu sistem yang mencerminkan eksekusi dari suatu aksi.



Gambar 2.9. *Action* pada *activity* diagram

1. *Start State*

*Start state* merupakan komponen yang digunakan dalam pembuatan *activity* diagram yang digunakan untuk mendeskripsikan bagaimana suatu objek dibentuk dan diawali.



Gambar 2.10. *Start state* pada *activity* diagram

1. *End State*

*End state* merupakan komponen yang digunakan dalam pembuatan *activity* diagram yang digunakan untuk mendeskripsikan bagaimana suatu objek dibentuk dan dihancurkan.



Gambar 2.11. *End state* pada *activity* diagram

1. *State Transition*

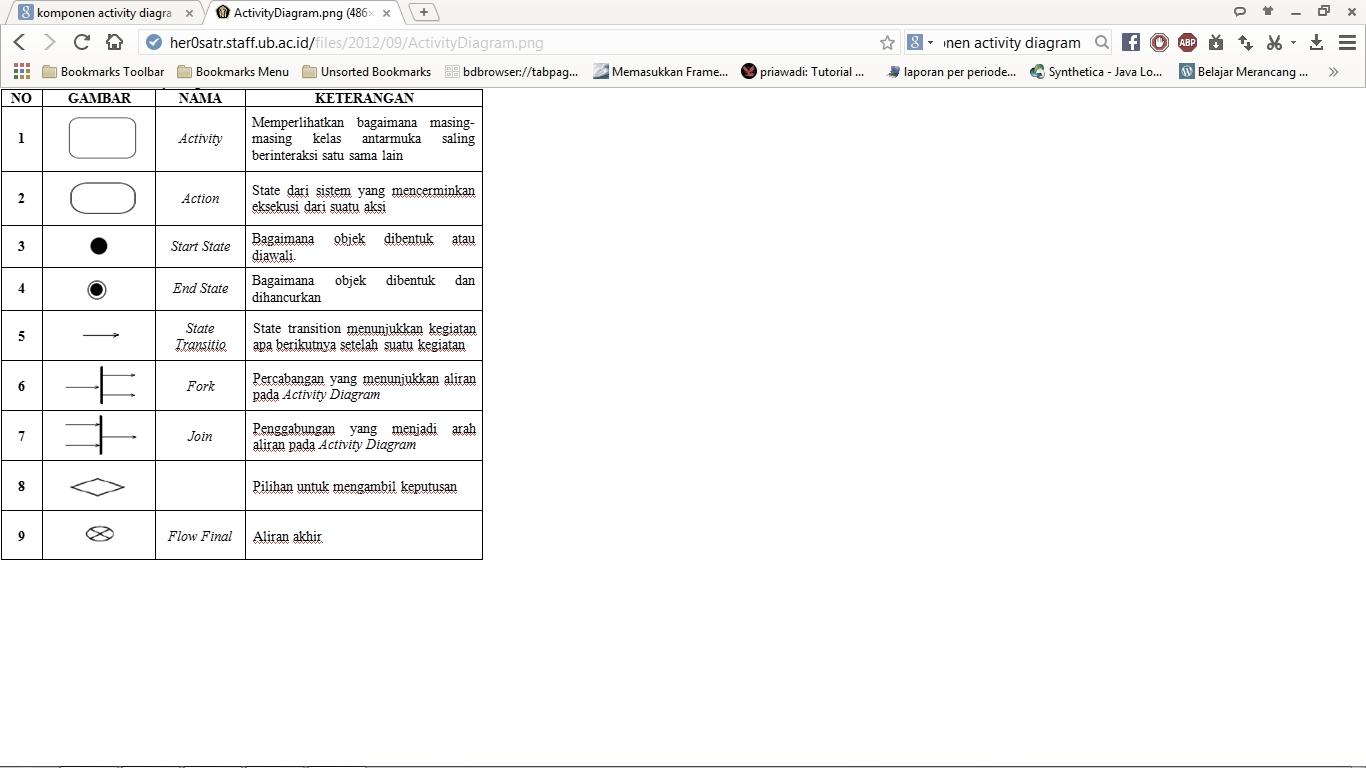
*State transition* merupakan komponen yang digunakan dalam pembuatan *activity* diagram yang digunakan untuk memperlihatkan atau menunjukkan kegiatan selanjutnya setelah suatu kegiatan telah selesai dilakukan.



Gambar 2.12. *State transition* pada *activity* diagram

1. *Fork*

*Fork* merupakan komponen yang digunakan dalam pembuatan *activity* diagram yang digunakan untuk menunjukkan aliran percabangan pada permodelan ini.



Gambar 2.13. *Fork* pada *activity* diagram

1. *Decision*

*Decision* merupakan komponen yang digunakan dalam pembuatan *activity* diagram yang digunakan untuk menunjukkan pilihan dalam mengambil sebuah keputusan.



Gambar 2.14. *Decision* pada *activity* diagram

1. *Flow Final*

*Flow final* merupakan komponen yang digunakan dalam pembuatan *activity* diagram yang digunakan untuk menunjukkan aliran akhir.



Gambar 2.15. *Flow Final* pada *activity* diagram

Berdasarkan beberapa pendapat yang dikemukakan diatas dapat ditarik kesimpulan bahwa “*Unified Modeling Language* (UML) adalah sebuah bahasa yang berdasarkan grafik atau gambar untuk menvisualisasikan, menspesifikasikan, membangun dan pendokumentasian dari sebuah sistem pengembangan perangkat lunak berbasis Objek / *Object Oriented programming (OOP*)”.