BAB II

TINJAUAN PUSTAKA

II.1. Konsep Dasar Sistem

Sistem merupakan kumpulan dari unsur atau elemen - elemen yang saling berkaitan/berinteraksi dan saling memengaruhi dalam melakukan kagiatan bersama untuk mencapai suatu tujuan tertentu (Asbon Hendra; 2012: 157).

II.1.1. Syarat - Syarat Sistem

Adapun beberapa syarat - syarat sistem yaitu sebagai berikut :

1. Sistem harus dibentuk untuk menyelesaikan tujuan.
2. Elemen sistem harus mempunyai rencana yang ditetapkan.
3. Adanya hubungan di antara elemen sistem.
4. Unsur dasar dari proses (arus informasi, energi, dan material) lebih penting dari pada elemen sistem.
5. Tujuan organisasi lebih penting dari pada tujuan elemen (Asbon Hendra; 2012: 158).

II.1.2. Karakteristik Sistem

Ada beberapa karakteristik dalam sistem diantaranya adalah :

1. Komponen

Suatu sistem terjadi dari sejumlah komponen yang saling berinteraksi, bekerja sama membentuk satu kesatuan.

2. Batas Sistem

Batas sistem merupakan daerah yang membatsai antara suatu sistem dengan sistem yang lainnya atau dengan lingkungan luarnya.

3. Lingkungan Luar Sistem

Lingkungan luas sistem merupakan segala sesuatu di luar batas sistem yang memengaruhi operasi dari suatu sistem. Lingkungan luar sistem ini dapat bersifat menguntungkan atau merugikan.

4. Penghubung Sistem

Penghubung sistem merupakan media penghubung antara satu subsistem dengan susbsitem yang lainnya untuk membentuk satu kesatuan sehingga sumber - sumber daya mengalir dari subsistem yang satu ke subsistem yang lainnya. Dengan kata lain, *output* dari suatu subsistem akan menjadi *input* dari subsistem yang lainnya.

5. Masukan Sistem *(Input)*

Input merupakan energi yang dimasukkan ke dalam sistem. Masukan dapat berupa Masukan Perawatan adalah energi yang dimasukkan supaya sistem tersebut dapat beroperasi.

6. Keluaran Sistem *(Output)*

Output merupakan hasil dari energi yang diolah oleh sistem, meliputi *output* yang berguna, Contohnya informasi yang dikeluarkan oleh komputer. Dan *output* yang tidak berguna dikenal sebagai sisa pembuangan, contohnya panas yang dikeluarkan oleh komputer.

7. Pengolah Sistem

Pengolah sistem merupakan bagian yang memproses masukan untuk menjadi keluaran yang diinginkan. Contoh CPU pada komputer, bagian produksi yang mengubah bbahan baku menjadi barang jadi, serta bagian akuntansi yang mengolah data transaksi menjadi laporan keuangan.

8. Tujuan Sistem

Setiap sistem pasti merupakan tujuan ataupun sasaran yang memengarungi *input* yang dibutuhkan dan *output* yang dihasilkan. Dengan kata lain, suatu sistem akan dikatakan berhasil kalau pengoperasian sistem itu mengenai sasaran atau tujuannya (Asbon Hendra; 2012: 158).

II.1.3. Klasifikasi Sistem

Dalam sistem informasi ada beberapa klasifikasi sistem diantaranya adalah :

1. Sistem Abstrak

Sistem yang berupa pemikiran atau ide - ide yang tidak tampak secara fisik. Sebagai contoh, sistem Teologia merupakan suatu sistem yang menggambarkan hubungan Tuhan dengan manusia.

2. Sistem Fisik

Merupakan sistem yang ada secara fisik sehingga setiap makhluk dapat melihatnya. Contohnya, sistem komputer, sistem akuntansi, sistem produksi, dan lain - lain.

3. Sistem Alamiah

Sistem yang terjadi melauli proses alam, dalam artian tidak dibuat oleh manusia, seperti sistem tata surya, sistem galaksi, sistem reproduksi, dan lain - lain.

II.2. Konsep Dasar Informasi

Informasi merupakan data yang telah diproses menjadi bentuk yang memiliki arti bagi penerima dan dapat berupa fakta, suati nilai yang bermanfaat. Jadi, ada suatu proses transformasi data menjadi suatu informasi = *input* – proses – *output.*

Data merupakan *raw material* untuk suatu informasi. Perbedaan informasi dan data sangat relatif, tergantung pada nilai gunanya bagi manajemen yang memerlukan. Suatu informasi bagi level manajemen tertentu bisa menjdi data bagi manajemen level di atasnya, atau sebaliknya (Asbon Hendra; 2012: 167).

II.2.1. Kualitas Informasi

Kualitas informasi merupakan satuan ukuran informasi, tergantung representasi. Untuk representasi biner, satuannya bit, byte, word, dan lain - lain.

Kualitas informasi tergantung dari tiga hal, yaitu informasi harus :

1. Akurat, berarti informasi harus bebas dari kesalahan - kesalahan dan tidak bias atau menyesatkan. Akurat juga berarti informasi harus jelas mencerminkan maksudnya.

2. Tepat pada waktunya, berarti informasi yang datang pada penerima tidak boleh terlambat.
3. Relevan, berarti informasi tersebut mempunyai manfaat untuk pemakainnya. Relevansi informasi untuk tiap - tiap orang satu dengan yang lainnya berbeda.

II.3. Sistem Pakar

Sistem pakar adalah sistem komputer yang ditujukan untuk meniru semua aspek *(emulates)* kemampuan pengambilan keputusan *(decision making)* seorang pakar. Sistem pakar memanfaatkan secara maksimal pengetahuan khusus selayaknya seorang pakar untuk memecahkan masalah (Rika Rosnelly; 2012: 2).

Sistem pakar merupakan salah satu aplikasi pertama yang muncul dari riset awal dalam bidang kecerdasan buatan. Penjelasan dari penalaran sistem pakar merupakan salah satu aplikasi pertama dari generasi bahasa alami. Hal ini disebabkan oleh kebutuhan untuk penjelasan adalah nyata, generasi dari aplikasi basis pengetahuan seperti penalaran harus secara relatif dan langsung (Rika Rosnelly; 2012: 105).

II.3.1. Kelebihan Sistem Pakar

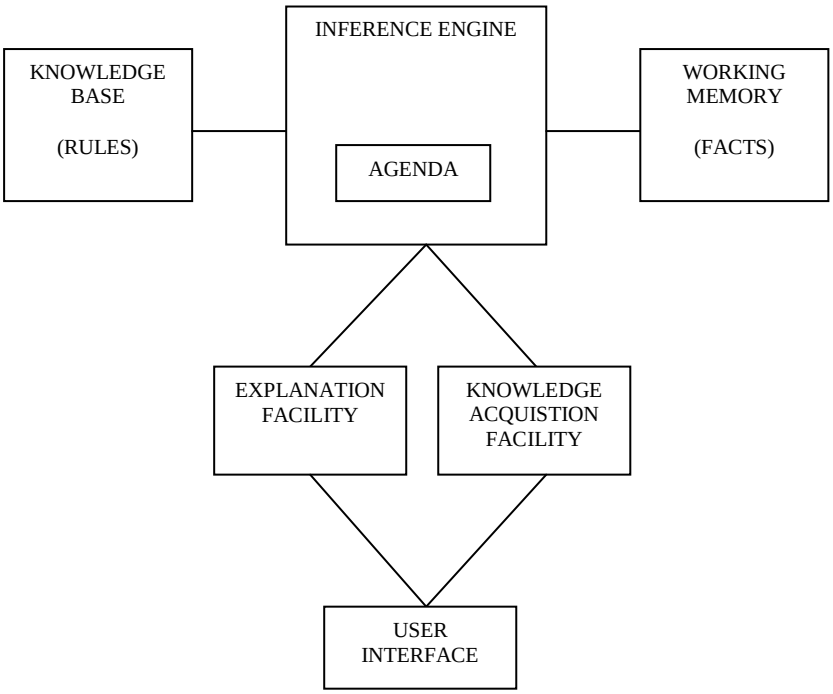
Sistem pakar memiliki beberapa fitur menarik yang merupakan kelebihannya, seperti :

1. **Meningkatkan ketersediaan.** Kepakaran atau keahlian menjadi tersedia dalam sistem komputer. Dapat dikatakan bahwa sistem pakar merupakan produksi kepakaran secara masal.

2. **Mengurangi biaya.** Biaya yang diperlukan untuk menyediakan keahlian satu per satu orang *user* menjadi berkurang.
3. **Mengurangi bahaya.** Sistem pakar dapat digunakan di lingkungan yang mungkin berbahaya bagi manusia.
4. **Permanen.** Sistem pakar dan pengetahuan yang terdapat didalamnya bersifat lebih permanen dibandingkan manusia yang dapat merasa lelah, bosan, dan pengetahuannya hilang saat sang pakar meninggal dunia.
5. **Keahlian multipel.** Pengetahuan dari beberapa pakar dapat dimuat ke dalam sistem dan bekerja secara simultan dan kontinyu menyelesaikan suatu masalah setiap saat. Tingkat keahlian atau pengetahuan yang digabungkan dari bebarapa pakar dapat melebihi pengetahuan satu orang pakar.
6. **Meningkatkan kehandalan.** Sistem pakar meningkatkan kepercayaan dengan memberikan hasil yang benar sebagai alternatif pendapat dari seseorang pakar atau sebagai penengah jika terjadi konflik antara beberapa pakar. Namun hal tersebut berlaku jika sistem dibuat oleh salah seorang pakar, sehingga akan selalu sama dengan pendapat pakar tersebut kecuali jika sang pakar melakukan kesalahan yang mungkin terjadi pada saat tertekan atau stres.
7. **Penjelasan.** Sistem pakar dapat menjelaskan detail proses penalaran yang dilakukan hingga mencapai suatu kesimpulan. Seorang mungkin saja terlalu lelah, tidak bersedia atau tidak mampu melakukannya setiap waktu. Hal ini akan meningkatkan tingkat kepercayaan bahwa kesimpulan yang dihasilkan adalah benar (Rika Rosnelly; 2012: 5).

II.3.2. Struktur Sistem Pakar

Adapun struktur sistem pakar dapat dilihat pada gambar II.1.



Gambar II.1. ***Struktur Sistem Pakar***
(Sumber : Rika Rosnelly; 2012: 13)

Komponen yang terdapat dalam struktur sistem pakar ini adalah :

1. *Knowledge Base* (Basis Pengetahuan)

Basis pengetahuan mengandung pengetahuan untuk pemahaman, formulasi, dan penyelesaian masalah. Komponen sistem pakar disusun atas dua elemen dasar, yaitu fakta dan aturan. Fakta merupakan informasi tentang objek dalam area permasalahan tertentu, sedangkan aturan merupakan informasi tentang cara bagaimana memperoleh fakta baru dari fakta yang telah diketahui. Pada struktur sistem pakar diatas, *knowledge base* disini untuk menyimpan pengetahuan dari

pakar berupa rule/aturan (if <kondisi> then <aksi> atau dapat juga disebut conditional-action rules).

2. *Inference Engine* (Mesin Inferensi)

Mesin inferensi merupakan otak dari sebuah sistem pakar dan dikenal juga dengan sebuah struktur control atau dalam sistem pakar berbasis kaidah. Komponen ini mengandung mekanisme pola pikir dan penalaran yang digunakan oleh pakar dalam menyelesaikan suatu masalah. Mesin inferensi disini adalah *processor* pada sistem pakar yang mencocokan bagian kondisi rule yang tersimpan di dalam *knowledge base* dengan fakta yang tersimpan di *working memory*.

3. *Working Memory*

Berguna untuk menyimpan fakta yang dihasilkan oleh *inference engine* dengan penambahan parameter berupa derajat kepercayaan atau juga dikatakan sebagai global database dari fakta yang digunakan oleh rule - rule yang ada.

4. *Explanation Facility*

Menyediakan kebenaran dari solusi yang dihasilkan kepada user *(reasoning chain).*

5. *Knowledge Acquisition Facility*

Meliputi proses pengumpulan, pemindahan dan perubahan dari kemampuan pemecahan masalah seorang pakar atau sumber pengetahuan terdokumentasi ke program komputer, yang bertujuan untuk memperbaiki atau mengembangkan basis pengetahuan.

6. *User Interface*

Mekanisme untuk memberi kesempatan kepada user dan sistem pakar untuk berkomunikasi. Antar muka menerima informasi dari pemakai dan mengubahnya ke dalam bentuk yang dapat diterima oleh sistem. Selain itu antarmuka menerima informasi dari sistem dan menyajikannya ke dalam bentuk yang dapat dimengerti oleh pemakai.

II.3.3. Karakteristik Sistem Pakar

Sistem pakar pada umunya dirancang untuk memenuhi beberapa karakteristik umum berikut ini :

1. **Kinerja sangat baik.** Sistem harus mampu memberikan respon berupa saran dengan tingkat kualitas yang sama dengan seorang pakar atau melebihinya.
2. **Waktu respon yang baik.** Sistem juga harus harus bekerja dalam waktu yang sama baiknya atau lebih cepat dibandingkan dengan seorang pakar dalam menghasilkan keputusan. Hal ini sangat penting terutama pada sistem waktu nyata.
3. **Dapat diandalkan.** Sistem harus dapat diandalkan dan tidak mudah rusak.
4. **Dapat dipahami.** Sistem harus mampu menjelaskan langkah - langkah penalaran yang dilakukannya seperti seorang pakar. Hal ini penting untuk beberapa alasan, yaitu :
   a. Dimungkinkan bahwa sistem pakar berkaitan dengan nyawa manusia atau properti lainnya harus dapat menjelaskan mengapa dihasilkan suatu kesimpulan tertentu.

b. Untuk mengkonfirmasi bahwa pengetahuan pakar telah disimpulkan dengan benar dan digunakan oleh sistem dengan benar pula. Hal ini penting dalam proses debugging pengetahuan yang mungkin salah karena pengetikan atau pemahaman yang salah dari *knowledge engineer*.

5. **Fleksibel.** Sistem harus menyediakan mekanisme untuk menambah, mengubah, dan menghapus pengetahuan (Rika Rosnelly; 2012: 20).

II.4. Faktor Kepastian *(Certainty Factor)*

Faktor Kepastian *(Certainty Factor)* adalah salah satu metode yang menunjukkan ukuran kepastian terhadap suatu fakta atau aturan. Notasi Faktor Kepastian : CF[h,e] = MB[h,e]-MD[h,e]

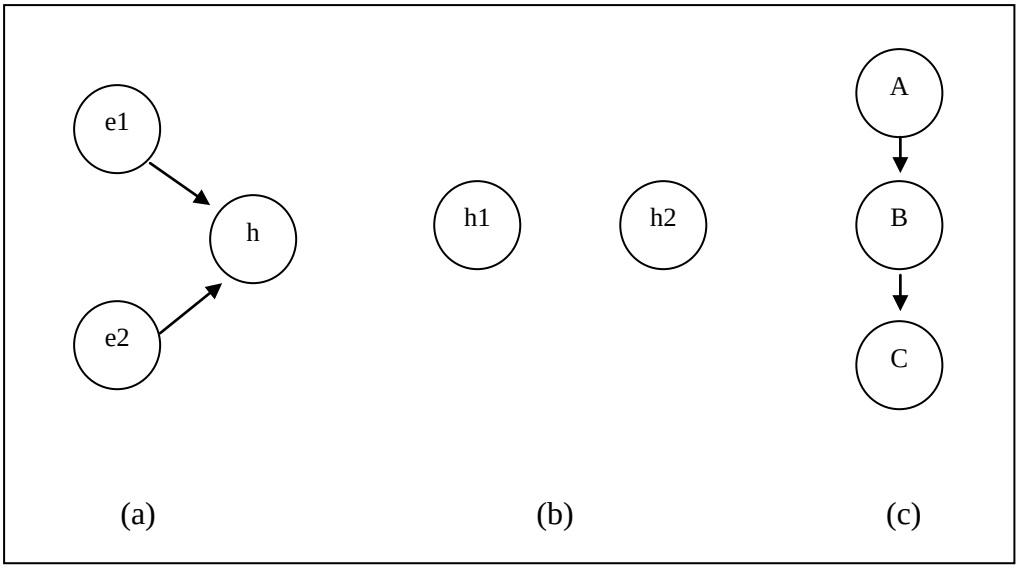
Dengan :

CF[h,e] = faktor kepastian.

MB[h,e] = ukuran kepercayaan terhadap hipotesis h, jika diberikan *evidence* e (antara 0 dan 1).

MD[h,e] = ukuran ketidakpercayaan terhadap hipotesis h, jika diberikan *evidence* e (antara 0 dan 1) .

Ada 3 hal yang mungkin terjadi :



Gambar II.2. ***Kombinasi aturan Ketidakpastian***
(Sumber : Rika Rosnelly; 2012: 89)

Beberapa *evidence* dikombinasikan untuk menentukan CF dari suatu hipotesis. Jika e1 dan e2 adalah observasi, maka :

MB[h,e1^e2] = 0 MD[h,e^e2] = 1

MB[h,e1] + MB[h,e2].(1-MB[h,e1])

MD[h,e1^e2] = 0 MB[h,e1^e2] = 1

MD[h,e1] + MD[h,e2].(1-MD[h,e1])

Contoh :

Si Ani menderita bintik - bintik di wajahnya. Dokter memperkirakan Si Ani terkena cacar dengan kepercayaan, MB[cacar, bintik - bintik] = 0,8 dan MD[cacar, bintik - bintik] = 0,01. Maka : CF[cacar, bintik - bintik] = 0,8 – 0,01 = 0,79.

Jika observasi tersebut juga memberikan kepercayaan bahwa Si Ani mungkin juga terkena alergi dengan kepercayaan, MB[alergi, bintik - bintik] = 0,4 dan MD[alergi, bintik - bintik] = 0,3. Maka : CF[alergi, bintik - bintik] = 0,4 – 0,3 = 0,1.

Untuk mencari CF[cacar ^ alergi, bintik - bintik] dapat diperoleh dari :

MB[cacar ^ alergi, bintik - bintik] = min(0,8;0,4) = 0,4

MD[cacar ^ alergi, bintik - bintik] = min(0,0,1;0,3) = 0,01

CF[cacar ^ alergi, bintik - bintik] = 0,4 – 0,01 = 0,39

Untuk mencari CF[cacar ∨ alergi, bintik - bintik] dapat diperoleh dari :

MB[cacar ∨ alergi, bintik - bintik] = max(0,8;0,4) = 0,8

MD[cacar ∨ alergi, bintik - bintik] = max(0,0,1;0,3) = 0,3

CF[cacar ∨ alergi, bintik - bintik] = 0,8 – 0,3 = 0,5

Dari contoh diatas dapat dilihat bahwa, semula faktor kepercayaan bahwa Si Ani terkena cacar dari gejala munculya bintik - bintik di wajah adalah 0,79. Demikian pula faktor kepercayaan bahwa Si Ani terkena alergi dari gejala munculnya bintik - bintik di wajah adalah 0,1. Dengan adanya gejala yang sama mempengaruhi 2 hipotesis yang berbeda ini, memberikan faktor kepercayaan bahwa :

- Si Ani menderita cacar dan alergi = 0,39
- Si Ani menderita cacar atau alergi = 0,5

II.5. *Louder* (Alat Berat)

Louder adalah alat yang dilengkapi dengan *bucket* untuk memuat material ke dalam truk. *Bucket* digunakan untuk mengorek *sleg* (tahi besi) untuk menggerakan *bucket* dapat dengan *cable* atau *hydrauolic*.

Louder terbagi atas dua jenis, yaitu :

1. *Crawler Louder*

   Louder jenis ini menggunakan ban dari besi yang cocok digunakan pada daerah dengan kondisi medan berat dengan permukaan tanah yang tidak rata.

2. *Wheel Louder*

   Wheel Louder menggunakan ban karet sehingga memiliki mobilitas yang lebih tinggi dibandingkan dengan *Crawler Louder.*

II.5.1. Kelebihan dan Kekurangan *Louder* (Alat Berat)

Kelebihan *Louder* adalah mobilitasnya yang tinggi dan daerah pemuatan loading point lebih sempit dibanding dengan *track shovel* dan kerusakan permukaan loading point lebih kecil karena menggunakan ban karet.

Salah satu kekurangannya adalah dalam menempatkan muatan ke dalam *dump truck* kurang merata nahkan kadang - kadang bisa miring, walaupun faktor ini sangat dipengaruhi oleh skill operator.

II.5.2. Cara Kerja *Louder* (Alat Berat)

Louder bekerja dengan cara gerakan dasar pada *Bucket* dan cara membawa muatan untuk dimuatkan ke alat angkut atau alat yang lain. Gerakan *Bucket* yang

penting adalah menurunkan *Bucket* diatas permukaan tanah, mendorong ke depan (memuat/menggusur), mengangkat *Bucket*, membawa dan membuang muatan kedalam truk.

II.6. Java dan *MySQL*

II.6.1. Java

Java merupakan salah satu forum yang bertebaran di internet, serta sifatnya *open source*, juga tersedia library. Salah satu kelebihan java adalah *independent*, terhadap operating system sehingga kompabilitas merupakan keunggulan tersendiri dibanding kan dengan bahasa yang lain.

Pengembangan aplikasi yang menuntut pembuatan chart menggunakan java sangat mudah dilakukan karena adanya library, yaitu *JfreeChart* dan *Commons library*. *JfreeChart library* adalah salah satu dari bnyak library untuk membuat chart java yang powerful (Mulkan Syarief; 2012: 2).

II.6.2. *MySQL*

MySQL bekerja menggunakan *SQL Language* (*Structure Query Language).* Itu dapat diartikan bahwa *MySQL* merupakan standart penggunaan database di dunia untuk pengolahan data.

Adapun beberapa kelebihan yang dimiliki oleh *MySQL* yaitu :

1. Bersifat open source, yang memiliki kemampuan untuk dapat dikembangkan lagi.

2. Menggunakan bahasa *SQL Language* (*Structure Query Language),* yang merupakan standar bahasa dunia dalam pengolahan data.
3. Super performance dan reliable, tidak bisa diragukan, pemrosesan database-nya sangat cepat dan stabil.
4. Sangat mudah dipelajari.
5. Memiliki dukungan support pengguna *MySQL* (Agus Saputra; 2011: 5).

II.7. UML (*Unified Modeling Languange)*

II.7.1. Pengenalan UML (*Unified Modeling Languange)*

UML *(Unified Modeling Language)* adalah suatu alat untuk memvisualisasikan dan mendokumentasikan hasil analisa dan desain yang berisi sintak dalam memodelkan sistem secara visual.

Sejarah UML sendiri terbagi dalam dua fase, sebelum dan sesudah munculnya UML. Dalam fase sebelum, UML sebenarnya sudah mulai diperkenalkan sejak tahun 1990an namun notasi yang dikembangkan oleh para ahli analisis dan desain berbeda - beda, sehingga dapat dikatakan belum memiliki standarisasi.

Fase kedua, dilandasi dengan pemikiran untuk mempersatukan metode tersebut dan dimotori oleh *Object Management Group (OMG)* maka pengembangan UML dimulai pada akhir tahun 1994 ketika Grady Booch dengan metode *Object Oriented Design (OOD).*

Saat ini sebagian besar para perancang sistem informasi dalam menggambarkan informasi dengan memanfaatkan UML diagram dengan tujuan

utama untuk membantu tim proyek berkomunikasi, mengeksplorasi potensi desain, dan memvalidasi desain arsitektur perangkat lunak atau pembuat program (Jurnal Informatika Mulawarman; 2011: 1).

II.7.2. Tujuan Pemanfaatan UML (*Unified Modeling Languange)*

Berikut tujuan utama dalam desain UML adalah :

1. Menyediakan bagi pengguna (analisis dan desain sistem) suatu bahasa pemodelan visual yang ekspresif sehingga mereka dapat mengembangkan dan melakukan pertukaran model data yang bermakna.
2. Menyediakan mekanisme yang spesialisasi untuk memperluas konsep inti.
3. Karena merupakan bahasa pemodelan visual dalam proses pembangunannya maka UML bersifat independen terhadap bahasa pemograman tertentu.
4. Memberikan dasar formal untuk pemahaman bahasa pemodelan.
5. Mendorong pertumbuhan pasar terhadap penggunaan alat desain sistem yang berorientasi objek.
6. Mendukung konsep pembangunan tingkat yang lebih tinggi seperti kolaborasi, kerangka, pola dan komponen terhadap suatu sistem.
7. Memiliki integrasi praktik terbaik (Jurnal Informatika Mulawarman; 2011: 2).

II.7.3. Struktur Diagram

Menggambarkan elemen dari spesifikasi dimulai dengan kelas, obyek, dan hubungan mereka, dan beralih ke dokumen arsitektur logis dari suatu sistem. Struktur diagram dalam UML terdiri atas :

1. *Class Diagram*

Class diagram menggambarkan struktur statis dari kelas dalam sistem anda dan menggambarkan atribut, operasi dan hubungan antara kelas. Class diagram membantu dalam memvisualisasikan struktur kelas - kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak dipakai.

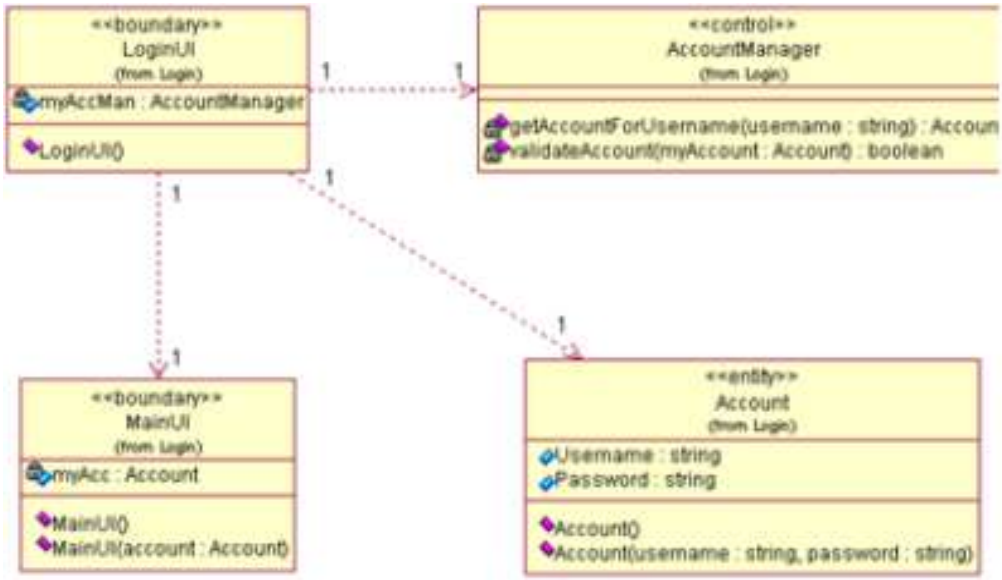
Class memiliki tiga area pokok :

a. Nama (dan stereotype).

b. Atribut.

c. Metoda.

Adapun gambar notasi *Class diagram* yaitu seperti dibawah ini:

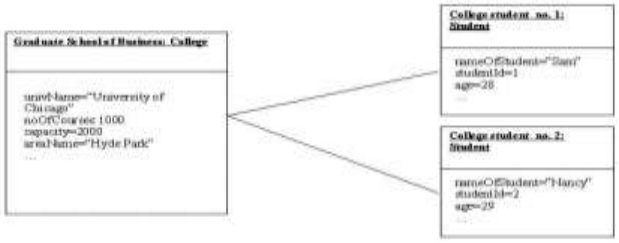


Gambar II.3. Notasi *Class diagram*
(Sumber : Jurnal Informatika Mulawarman; 2011: 3)

2. Object Diagram

Object diagram menggambarkan kejelasan kelas dan warisan dan kadang-kadang diambil ketika merencanakan kelas, atau untuk membantu pemangku kepentingan non-program yang mungkin menemukan diagram kelas terlalu abstrak.

Adapun gambar notasi *Objek diagram* yaitu seperti dibawah ini :

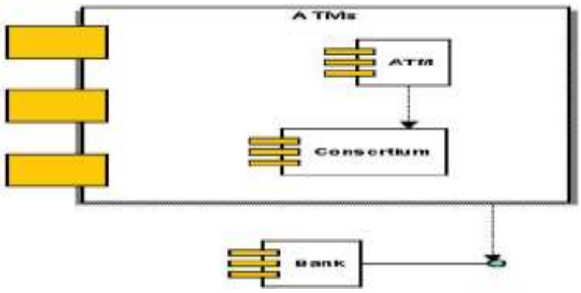


Gambar II.4. Notasi *Object diagram*
(Sumber : Jurnal Informatika Mulawarman; 2011: 3)

3. Component Diagram

Component diagram menggambarkan struktur fisik dari kode, pemetaan pandangan logis dari kelas proyek untuk kode aktual di mana logika ini dilaksanakan.

Adapun gambar notasi *Component diagram* yaitu seperti dibawah ini :

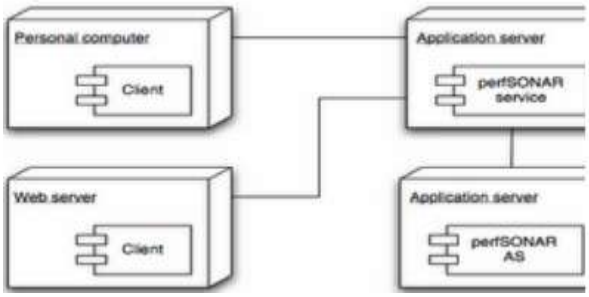


Gambar II.5. Notasi *Component diagram*
(Sumber : Jurnal Informatika Mulawarman; 2011: 3)

4. Deployment Diagram

Deployment diagram memberikan gambaran dari arsitektur fisik perangkat lunak, perangkat keras, dan artefak dari sistem. Deployment diagram dapat dianggap sebagai ujung spektrum dari kasus penggunaan menggambarkan bentuk fisik dari sistem yang bertentangan dengan gambar konseptual dari pengguna dan perangkat berinteraksi dengan system.

Adapun gambar notasi *deployment diagram* yaitu seperti dibawah ini :

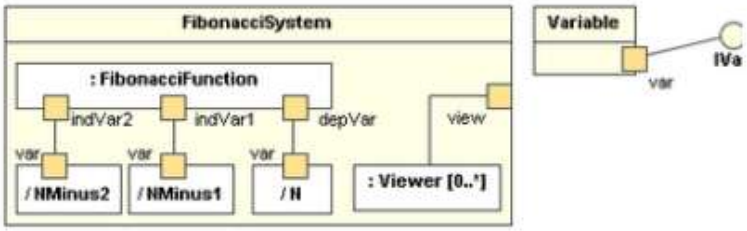


Gambar II.6. Notasi *deployment diagram*
(Sumber : Jurnal Informatika Mulawarman; 2011: 4)

5.Composite Structure Diagram

Composite structure diagram Sebuah diagram struktur komposit mirip dengan diagram kelas, tetapi menggambarkan bagian individu, bukan seluruh kelas. Kita dapat menambahkan konektor untuk menghubungkan dua atau lebih bagian dalam atau ketergantungan hubungan asosiasi.

Adapun gambar notasi *Composite structur diagram* yaitu seperti dibawah ini :

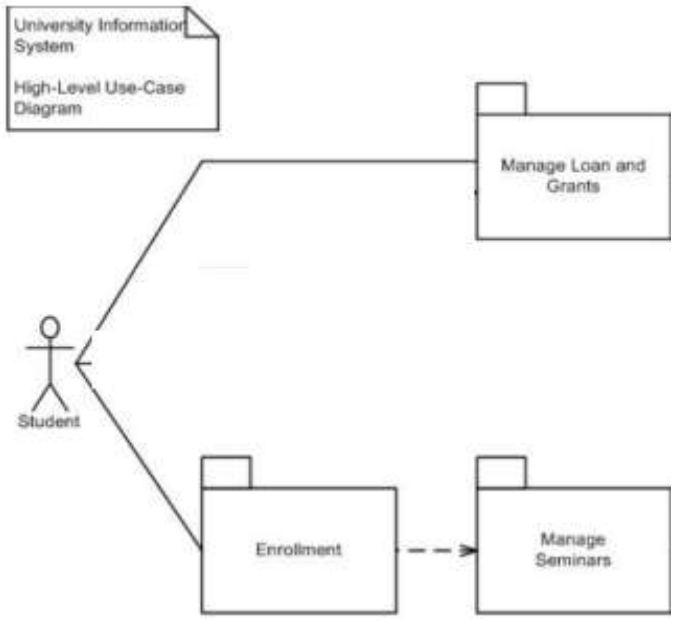


Gambar II.7. Notasi *Composite structure diagram*
(Sumber : Jurnal Informatika Mulawarman; 2011: 4)

6. Package Diagram

Paket diagram biasanya digunakan untuk menggambarkan tingkat organisasi yang tinggi dari suatu proyek software. Atau dengan kata lain untuk menghasilkan diagram ketergantungan paket untuk setiap paket dalam Pohon Model.

Adapun contoh gambar notasi *package diagram* yaitu seperti dibawah ini:

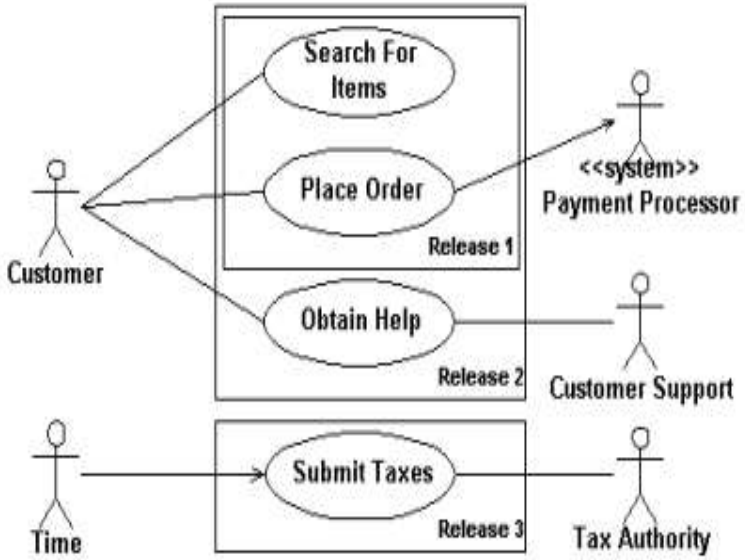


Gambar II.8. Notasi *Package diagram*
(Sumber : Jurnal Informatika Mulawarman; 2011: 4)

7. Use case Diagram

Diagram yang menggambarkan actor, use case dan relasinya sebagai suatu urutan tindakan yang memberikan nilai terukur untuk aktor. Sebuah *use case* digambarkan sebagai elips horizontal dalam suatu diagram UML *use case.*

Adapun contoh gambar notasi *use case diagram* yaitu seperti dibawah ini :

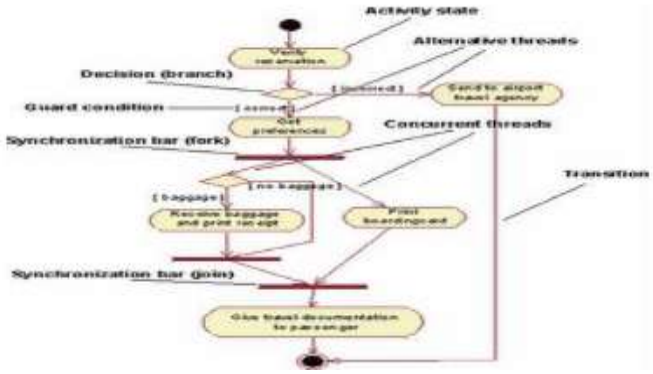


Gambar II.9. Notasi *use case diagram*
(Sumber : Jurnal Informatika Mulawarman; 2011: 4)

8. Activity Diagram

Menggambarkan aktifitas - aktifitas, objek, *state,* transisi *state* dan *event.* Dengan kata lain kegiatan diagram alur kerja menggambarkan perilaku sistem untuk aktivitas.

Adapun contoh gambar notasi *Activity diagram* yaitu seperti dibawah ini:

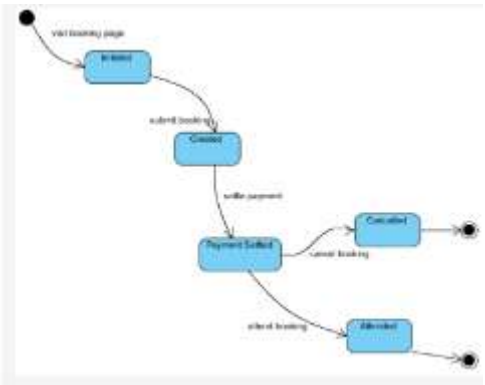


Gambar II.10. Notasi *Activity diagram*
(Sumber : Jurnal Informatika Mulawarman; 2011: 4)

9. State Machine diagram

Menggambarkan state, transisi state dan event.

Adapun contoh gambar notasi *State Mchine diagram* yaitu seperti dibawah ini :

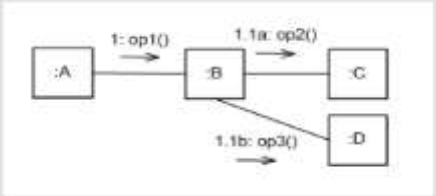


Gambar II.11. *State Machine diagram*
(Sumber : Jurnal Informatika Mulawarman; 2011: 5)

10. Communication Diagram

Serupa dengan *sequence* diagram, tetapi diagram komunikasi juga digunakan untuk memodelkan perilaku dinamis dari use case. Bila dibandingkan dengan *Sequence* diagram, diagram komunikasi lebih terfokus pada menampilkan kolaborasi benda daripada urutan waktu.

Adapun contoh gambar notasi *Communicatin diagram* yaitu seperti dibawah ini :

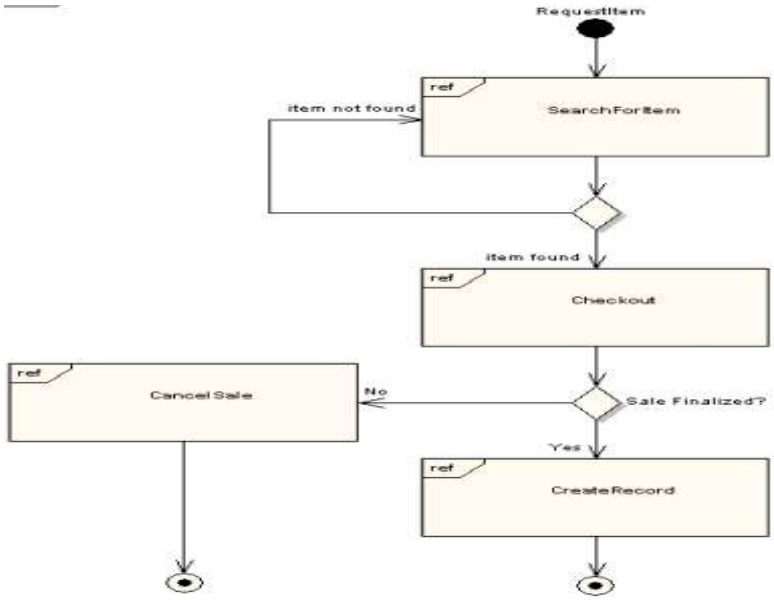


Gambar II.12. Notasi *Communication diagram*
(Sumber : Jurnal Informatika Mulawarman; 2011: 5)

11. Interaction Overview Diagram

Interaksi overview diagram berfokus pada gambaran aliran kendali interaksi dimana node adalah interaksi atau kejadian interaksi.

Adapun contoh gambar notasi *Interaction OverView diagram* yaitu seperti dibawah ini :

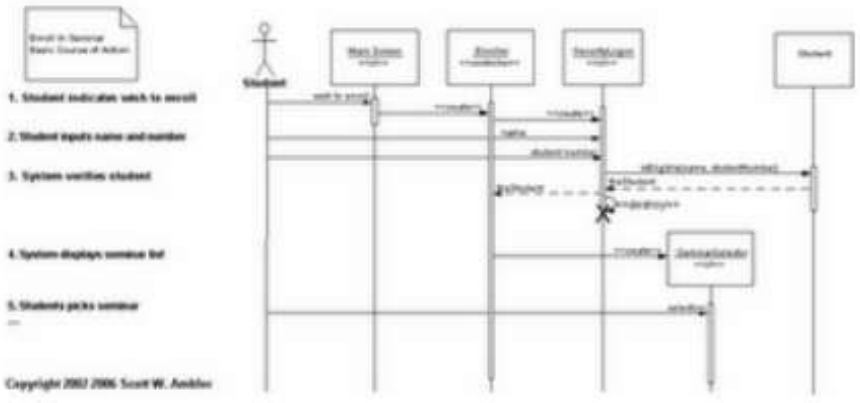


Gambar II.13. Notasi *Interaction Overview diagram*
(Sumber : Jurnal Informatika Mulawarman; 2011: 5)

12. Sequence Diagram

Sequence diagram menjelaskan interaksi objek yang disusun berdasarkan urutan waktu. Secara mudahnya sequence diagram adalah gambaran tahap demi tahap, termasuk kronologi (urutan) perubahan secara logis yang seharusnya dilakukan untuk menghasilkan sesuatu sesuai dengan *use case diagram.*

Adapun contoh gambar notasi *Sequence diagram* yaitu seperti dibawah ini:

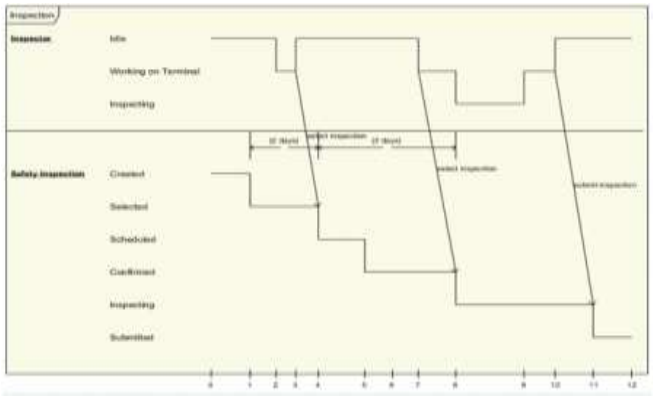


Gambar II.14. Notasi *Sequence Diagram*
(Sumber : Jurnal Informatika Mulawarman; 2011: 5)

13. Timing Diagram

Timing diagram di UML didasarkan pada diagram waktu *hardware* awalnya dikembangkan oleh para insinyur listrik.

Adapun contoh gambar notasi *Timing diagram* yaitu seperti dibawah ini :

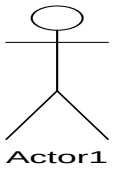


Gambar II.15. Notasi *Timing diagram*
(Sumber : Jurnal Informatika Mulawarman; 2011: 6)

II.7.4. Notasi - Notasi UML

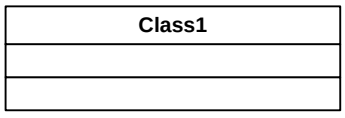
UML memiliki seperangkat notasi yang akan digunakan ke dalam tiga kategori diatas yaitu struktur diagram, behaviour diagram dan interaction diagram. Berikut beberapa notasi dalam UML diantaranya :

1. *Actor;* menentukan peran yang dimainkan oleh user atau sistem lain yang berinteraksi dengan subjek. *Actor* adalah segala sesuatu yang berinteraksi langsung dengan sistem aplikasi komputer, seperti orang, benda atau lainnya.



Gambar II.16. Notasi *actor* pada UML
(Sumber : Jurnal Informatika Mulawarman; 2011: 6)

2. *Class* diagram; Notasi utama dan yang paling mendasar pada diagram UML adalah notasi untuk mempresentasikan suatu class beserta dengan atribut dan operasinya. *Class* adalah pembentuk utama dari sistem berorientasi objek.



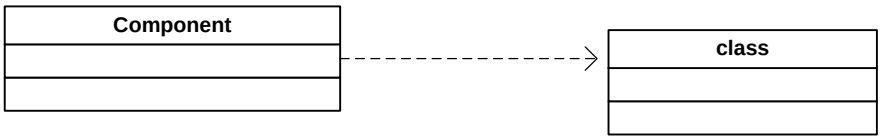
Gambar II.17. Notasi *class* pada UML
(Sumber : Jurnal Informatika Mulawarman; 2011: 6)

3. *Use case* adalah deskripsi fungsi dari sebuah sistem perspektif pengguna. *Use case* bekerja dengan cara mendeskripsikan tipikal interaksi antara

user (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai.

Gambar II.18. Notasi *Use case* pada UML
(Sumber : Jurnal Informatika Mulawarman; 2011: 6)

4. *Realization* menunjukkan hubungan bahwa elemen yang ada di bagian tanpa panah akan merealisasikan apa yang dinyatakan oleh elemen yang ada di bagian dengan panah.



Gambar II.19. Notasi *Realization* pada UML
(Sumber : Jurnal Informatika Mulawarman; 2011: 6)

5. *Interaction* digunakan untuk menunjukkan baik aliran pesan atau informasi antar obyek maupun hubungan antar obyek.

Gambar II.20. Notasi *Interaction* pada UML
(Sumber : Jurnal Informatika Mulawarman; 2011: 6)

6. *Dependency* merupakan relasi yang menunjukan bahwa perubahan pada salah satu elemen memberi pengaruh pada elemen lain. Terdapat 2 *stereotype* dari *dependency*, yaitu *include* dan *extend.* Include menunjukkan bahwa suatu bagian dari elemen (yang ada digaris tanpa panah) memicu eksekusi bagian dari elemen lain (yang ada di garis dengan panah).

Gambar II.21. Notasi *Dependency* pada UML
(Sumber : Jurnal Informatika Mulawarman; 2011: 6)

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

Gambar II.22. Notasi *Note* pada UML
(Sumber : Jurnal Informatika Mulawarman; 2011: 7)

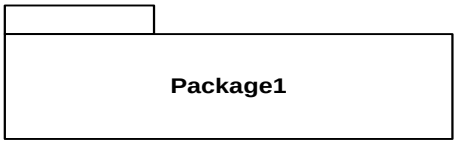
8. *Association* menggambarkan navigasi antar class (navigation), berapa banyak obyek lain yang bisa berhubungan dengan satu obyek (*multiplicity* antar *class*) dan apakah suatu class menjadi bagian dari class lainnya (aggregation).

Gambar II.23. Notasi *Association* pada UML
(Sumber : Jurnal Informatika Mulawarman; 2011: 7)

9. *Generalization* menunjukkan hubungan antara elemen yang lebih umum ke elemen yang lebih spesifik.

Gambar II.24. Notasi *Generalization* pada UML
(Sumber : Jurnal Informatika Mulawarman; 2011: 7)

10. *Package* adalah mekanisme pengelompokkan yang digunakan untuk menandakan pengelompokkan elemen - elemen model.



Gambar II.25. Notasi *Package* pada UML
(Sumber : Jurnal Informatika Mulawarman; 2011: 7)

11. *Interface* merupakan kumpulan operasi berupa implementasi dari suatu *class*. Atau dengan kata lain implementasi operasi dalam *interface* dijabarkan oleh operasi di dalam *class*.

Gambar II.26. Notasi *Interface* pada UML
(Sumber : Jurnal Informatika Mulawarman; 2011: 7)