BAB II

TINJAUAN PUSTAKA

Pada bab ini akan menjelaskan beberapa teori yang berkaitan dengan permasalahan yang akan dibahas sebagai dasar dalam pemahaman dalam sebuah sistem serta metoda yang dipakai untuk kegiatan pengembangan terhadap sistem itu sendiri.

2.1 Konsep Dasar Sistem

Suatu sistem mempunyai sifat-sifat dasar diantaranya :

a. Tujuan

   Setiap sistem mempunyai tujuan pasti dan jelas sehingga kegiatan sistem menjurus kepada tujuan.

b. Kesatuan

   Hasil dari suatu kesatuan akan lebih besar dari jumlah masing-masing bagian. Jika bekerja secara terpisah dan kesatuan ini harus menghasilkan optimasi yang memberikan manfaat terbaik terhadap sistem.

c. Interaksi

   Terdapat hubungan khusus dimana akan terjadi saling keterkaitan dan ketergantungan antara elemen-elemen pembentuk sistem dan antara sistem dengan lingkungannya.

d. Transformasi

   Suatu sistem memiliki daya menciptakan dan meningkatkan nilai guna dengan jalan memanfaatkan dan merubah sumber-sumber yang ada menjadi *output* melalui suatu proses.

e. Keterbukaan

Sistem mempunyai keterbukaan terhadap pengaruh lingkungan dimana sumber dan pemakai nilai-nilai yang dihasilkan sistem tersebut berbeda.

f. Mekanisme Kontrol

Sistem merupakan suatu rangkaian sehingga memungkinkan terdapatnya suatu proses pemberian umpan balik dan kemampuan mengatur dirinya sendiri untuk dapat mempertahankan eksistensinya dan menyeimbangkan dirinya ke suatu posisi yang lain.

2.2 Pengertian Sistem

Beberapa ahli sistem mendefinisikan sistem sebagai berikut :

Menurut JOG [1], sistem adalah :

"Suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama untuk melakukan kegiatan dan atau menyelesaikan suatu sasaran tertentu." JOG [1]

Menurut ZUL [2],berdasarkan definisi dari kamus Webster's Unabridged, sistem adalah :

"Elemen-elemen yang saling berhubungan membentuk satu kesatuan atau organisasi." ZUL [2]

Dari beberapa definisi tersebut, penulis menyimpulkan, sistem adalah rangkaian elemen-elemen organisasi yang membentuk sebuah prosedur kerja yang berkumpul dan bersatu untuk menyelesaikan tujuan tertentu..

2.3 Klasifikasi Sistem

Sistem dapat diklasifikasikan dari beberapa sudut pandangan, menurut JOG [1],mengklasifikasikan sistem sebagai berikut :

1. Sistem Abstrak dan Sistem Fisik

   Sistem abstrak adalah sistem yang berupa pemikiran atau ide-ide yang tidak tampak secara fisik seperti sistem teologia. Sedangkan sistem fisik merupakan sistem yang ada secara fisik. seperti sistem komputer.

2. Sistem Alamiah dan Sistem Buatan Manusia

   Sistem alamiah adalah sistem yang terjadi melalui proses alam, tidak dibuat oleh manusia seperti sistem perputaran bumi. Sedangkan sistem buatan manusia adalah sistem yang dirancang oleh manusia.

3. Sistem Tertentu dan Sistem Tak Tentu

   Sistem tertentu beroperasi dengan tingkah laku yang sudah dapat diprediksi. Sistem komputer adalah contoh dari sistem tertentu. Sedangkan sistem tak tentu adalah sistem dimana kondisi masa depan tidak dapat diprediksi karena mengandung unsur probabilitas.

4. Sistem Tertutup dan Sistem Terbuka

   Sistem tertutup merupakan sistem yang tidak berhubungan dan tidak terpengaruh oleh lingkungan luar. Sedangkan sistem terbuka adalah sistem yang berhubungan dan terpengauh dengan lingkungan luar.

2.4 Pengertian Informasi

Menurut JOG [8], informasi dapat diidentifikasikan sebagai data yang telah diolah menjadi suatu bentuk yang penting bagi yang menerimanya. Sumber informasi adalah data. Jadi informasi adalah data yang telah diolah yang mempunyai nilai guna atau manfaat bagi si pemakai dalam proses pengambilan keputusan atau informasi atau *output* dari proses transformasi dimana data tersebut berfungsi sebagai *input.*

Kualitas dari informasi tergantung dari beberapa hal, yaitu:

1. Keakuratan, dapat didefinisikan sebagai perbandingan dari informasi yang benar dengan jumlah seluruh informasi yang dihasilkan pada satu proses pengolahan data tertentu. Berarti informasi harus bebas dari kesalahan-kesalahan dan tidak menyesatkan.
2. Tepat waktu, berarti informasi yang datang pada penerima tidak boleh terlambat. Karena informasi yang sudah usang tidak akan bernilai lagi. Informasi merupakan landasan bagi pengambilan keputusan maka jika terjadi keterlambatan dapat berakibat fatal bagi perusahaan atau organisasi itu sendiri.
3. Relevan, berarti informasi tersebut mempunyai manfaat untuk pemakainya. Tingkat relevansi informasi untuk tiap berbeda-beda.

Dari uraian data di atas dapat disimpulkan bahwa sistem informasi adalah sekelompok manusia, sekumpulan petunjuk dan peralatan pengolah data untuk memilih, menyimpan, pengolahan dan pencarian data untuk mengurangi ketidakpastian dalam pengambilan keputusan dengan menyediakan informasi bagi pimpinan pada waktu dimana mereka dapat menggunakannya secara efisien. Tujuan dari sistem informasi adalah menciptakan suatu sistem yang mampu menyediakan informasi-informasi yang terstruktur dan sistematis untuk mendukung para pemimpin fungsi-fungsinya sehingga tercapai suatu keputusan yang didasarkan pada pendekatan sistem.

2.5 Pengertian Sistem Informasi

Menurut JOG [1], sistem informasi adalah :

"Suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan yang diperlukan." [JOG:1]

Penulis dapat menyimpulkan bahwa sistem informasi adalah sistem yang menyediakan kebutuhan untuk proses operasi dan manajerial suatu organisasi serta membuatkannya ke dalam sebuah laporan yang diperlukan bagi pihak luar organisasi.

2.6 Metode Pengembangan Sistem

Metode adalah suatu cara / teknik sistematis untuk mengerjakan sesuatu. Metodologi adalah metode-metode, prosedur-prosedur, konsep-konsep pekerjaan, aturan-aturan yang akan digunakan sebagai pedoman bagaimana dan apa yang harus dikerjakan selama proses pengembangan sistem.

Penulis mengambil model prototipe sebagai model pengembangan sistem. Model prototipe (prototyping model), merupakan suatu teknik untuk mengumpulkan informasi tertentu mengenai kebutuhan-kebutuhan informasi pengguna secara cepat. Tahapan yang dilakukan :

1 Reaksi awal dari pengguna, diawali dengan menampilkan sebuah prototipe sistem informasi, kemudian melihat reaksi dari pengguna saat bekerja dengan prototipe apakah fitur-fitur sistem pada prototipe tersebut sudah sesuai dengan kebutuhannya. Reaksi tersebut dikumpulkan dalam lembar observasi, wawancara dan quesioner.

2 Saran-saran pengguna, saran-saran merupakan hasil interaksi pengguna dengan prototipe yang ditampilkan (evaluasi pengguna) yang merupakan masukan untuk perbaikan, pengubahan atau menghentikan prototipe sehingga dapat memenuhi kebutuhan pengguan dengan lebih baik.

3 Inovasi, adalah kemampuan-kemampuan sistem baru yang sebelumnya tidak ada pada saat pengguna berinteraksi dengan prototipe. Inovasi prototipe jika berhasil akan menjadi bagian dari sistem hasil jadi.

Rencana revisi, prototipe menggambarkan sistem di masa datang. Rencana revisi membantu mengidentifikasikan prioritas-prioritas apa saja yang akan diprototipekan selanjutnya.

2.7 Unified Modeling Language (UML)

2.7.1 Pengertian Unified Modeling Language (UML)

Menurut Hans-Erik **Unified Modelling Language** (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. UML mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (Object-Oriented Design), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*).

UML Menggabungkan konsep berarah objek seperti Booch, OMT, OOSE, dsb sehingga menjadi bahasa pemodelan tunggal. UML menekankan pada apa yang dapat dikerjakan dengan metode-metode tersebut. UML berfokus pada suatu bahasa pemodelan standar bukan pada proses standar.

2.7.2 Bagian - bagian dari UML

Menurut Hans-Erik bagian–bagian utama dari UML adalah *view, diagaram, model element, dan general mechanism.*

1. View

View digunakan untuk melihat sistem yang dimodelkan dari beberapa aspek yang berbeda. *View* bukan melihat gratifk, tapi merupakan suatu abstraksi yang berisi sejumlah diagram. Beberapa jenis view dalam UML antarai lain : *use case view, logical viw , component viwm , concurrences view,* dan *deployment view*.

a. Use case view

Mendeskripsikan fungsionalitas sistem yang seharusnya dilakukan sesuai yang diinginkan *external actors. Actor* yang berinterakasi dengan sistem dapat berupa user atau sistem lainnya. *View* ini digambarkan dalam *use case diagrams* dan kadang–kadang dengan *activity diagramsView* ini digunakan terutama untuk pelanggan , perancang (*designer*) , pengembang *(developer*), dan penguji sistem (*tester*).

b. Logical view

Mendeskripsikan bagaimana fungsionalitas dari sistem, struktur statis (*class , object*, dan r*elationship*) dan kolaborasi dinamis yang terjadi ketika *object* mengirim pesan ke *object* lain dalam suatu fungsi tertentu. View ini

digambarkan dalam *class diagrams* untuk struktur statis dan dalam *state sequence*, *collaboration, dan activity diagrams* untuk model dinamisnya, *View* ini digunakan untuk perancang (designer) dan pengembang (developer).

c. Component view

Mendiskripsikan implementasi dan ketergantungan modul. Komponen yang merupakan tipe lainnya dari *code module* diperlihatkan dengan struktur dan ketergantungannya juga alokasi sumber daya komponen dan informasi administrative lainnya. *View* ini diganbarkan dalam *component view* dan digunakan untk pengembang (developer).

d. Concurrency view

Membagi sistem ke dalam proses dan prosessor. *View* ini digambarkan dalam diagram dinamis (*state, sequence, collaboration*, dan *activity diagrams*) dan diagram implementasi (*component* dan *deployment diagrams*) serta digunakan untk pengembang (developer), pengintegrasi (integrator), dan penguji (tester).

e. Deployment view

Mendeskripsikan fisik dari sistem seperti komputer dan perangkat (*nodes*) dan bagaimana hubungannya dengan lainnya. *View* ini digambarkan dalam deplovment diagrams dan digunakan untuk pengembang (developer), pengintegrasi (integrator), dan penguji (tester).

2. Diagram

Diagram berbentuk grafik yang menunjukan simbol elemen model yang disusun untuk mengilustrasikan bagian atau aspek tertentu dari sistem. Sebuah diagram merupakan bagian dari suatu *view* tertentu dan ketika digambarkan biasanya dialokasikan untuk *view* tertentu. Adapun jenis diagram antara lain :

a. Use Case Diagram

Menggambarkan sejumlah *external actors* dan hubungannya ke *use case* yang diberikan oleh sistem. *Use case* adalah deskirpsi fungsi yang disediakan oleh sistem dalam bentuk teks sebagai dokumentasi dari *use case symbo*l namun juga dilakukan dalam *activity diagrams*. *Use case* digambarkan hanya yang dilihat dari luar oleh *actor* (keadaan lingkungan sistem yang dilihat user) dan bukan bagaimana fungsi yang ada di dalam system.

b. Class Diagram

Menggambarkan struktur statis *class* di dalam sistem. *Class* merepresentasikan sesuatu yang ditangani oleh sistem. *Class* dapat berhubungan dengan yang lain melalui berbagai cara : *associated* (terhubunga satu sama lain), *dependent* (satu *class* tergantung / menggunakan *clas*s yang lain), *speciated* (satu *class* merupakan spesialisasi dari *class* lainnya), atau *package* (grup bersama sebagai satu unit). Sebuah sistem biasanya mempunyai beberapa *class diagrams.*

c. State Diagram

Menggambarkan semua *state* (kondisi) yang dimiliki oleh suatu *objec*t dari suatu *class* dan keadaan yang menyebabkan *state* berubah.. Kejadian dapat berupa *object* lain yang mengirim pesan. *State class* tidak digambarkan untuk semua *class*, hanya yang mempunyai jumlah *state* yang terdefinisi dengan baik dan kondisi *class* berubah oleh *state* yang berbeda.

d. Sequence Diagram

Menggambarkan kolaborasi dinamis antara sejumlah object. Kegunaannya untuk menunjukan rangkaian pesan yang dikirim antar *object* juga interaksi antara *object* sesuatu yang terjadi pada titik tertentu dalam eksekusi sistem.

e. Collaboration Diagram

Menggambarkan kolaborasi dinamis seperti *sequence diagrams*. Dalam menunjukan pertukaran pesan, *collaboration diagrams* menggambarkan *object* dan hubungannya (mengacu ke konteks). Jika penekannya pada waktu atau urutan digunakan *sequence diagrams*, tapi jika penekannya pada konteks gunakan *collaboration diagrams.*

f. Activity Diagram

Menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktivitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktivitas lainnya seperti *user case* atau interaksi.

g. Component Diagram

Menggambarkan struktur fisik kode dari kompen. Komponen dapat berupa *source code*, komponen biner, atau *executeable component*. Sebuah komponen berisi informasi tentang *logic class* atau *class* yang diimplementasikan sehingga menbuat pemetaan dari logical view ke *component view*.

h. Deployment Diagram

Menggambarkan arsitektur fisik dari perangkat keras dan perangkat lunak sistem, menunjukan hubungan komputer dengan perangkat *(nodes*) satu sama lain dan jenis hubungannya. Di dalam *nodes*, *executeable component* dan *object* yang dialokasikan untuk memperlihatkan unit perangkat lunak yang dieksekusi oleh *node* tertentu dan ketergantungan komponen.

2.7.3 Tujuan Pengunaan UML

1. Memodelkan suatu sistem (bukan hanya perangkat lunak) yang menggunakan konsep beorientasi *object*
2. Menciptakan suatu bahasa pemodelan yang dapat digunakan baik oleh manusia maupun mesin.

2.8 Komponen-Komponen Yang Terlibat Dalam Use Case Diagram

1. Actor

Menurut Hans-Erik pada dasarnya *actor* bukanlah bagian dari *use case diagram* , namun untuk dapat terciptanya suatu *use case diagram* diperlukan beberapa *actor* dimana *actor* tersebut mempresentasikan seseorang atau sesuatu (seperti perangkat, sistem lain) yang berinteraksi dengan sistem. Sebuah *actor* mungkin hanya memberikan informasi inputan pada sisstem, hanya menerima infromasi dari sistem atau keduannya menerima dan memberi informasi pada sistem, *actor* hanya berinterkasi dengan use case tetapi tidak memiliki kontrol atas *use case*. *Acttor* digambarkan dengan *stick man.*

Ada beberapa kemungkinan yang menyebabkan *actor* tersebut terkait dengan sistem antara lain :

a. Yang berkepentingan terhadap sistem dimana adanya arus informasi baik yang diterimanya maupun yang dia inputkan ke sistem
b. Orang ataupun pihak yang akan mengelola sistem tersebut
c. *External resource* yang digunkan oleh sistem
d. Sistem lain yang berinteraksi dengan sistem yang akan dibuat

2. Use Case

Use Case adalah gambaran fungsionalitas dari suatu sistem , sehingga *costumer* atau pengguna sistem paham dan mengerti mengenai kegunaan sistem yang akan di bangun. *Use case* diagram adalah penggambaran sistem dari sudut pandang pengguna sistem tersebut (*user*), sehingga pembuatan *use case* lebih dititikberatkan pada fungsionalitas yang ada pada sistem , bukan berdasarkan alur atau urutan kejadian. Cara menentukan Use Case dalam suatu sistem :

a. Pola perilaku perangkat lunak aplikasi
b. Gambaran tugas dari sebuah *actor*
c. Sistem atau "benda" yang memberikan sesuatu yang berniali kepada *actor*
d. Apa yang dikerjakan oleh suatu perangkat lunak (bukan bagaimana cara mengerjakannya).

3. Relasi dalam Use Case

Ada beberapa relasi yang terdapat pada *use case diagarm* :

a. *Association* , menghubungkan link antar element
b. *Generaliization* , disebut juga *inheritance* (pewarisan), sebuah element dapat merupakan spesialiasi dari elemen lainnya.
c. *Dependency*, sebuah element bergantung dalam beberapa cara elemen lainnya.
d. *Aggregation*, bentuk assosiation dimana sebuah elemen berisi elemen lainnya

4. Use Case Diagram

Adalah gambaran grapichal dari beberapa atau semua actorm use case dan interaksi diantaranya yang memperkenalkan suatu sistem.

2.9 Class Diagram

2.9.1 Definsi Object dan Class

Menurut Hans-Erik *Object* adalah gambaran dari entity, baik dunia nyata atau konsep dengan batasan – batasan dan pengertian yang tepat. *Objec*t bisa mewakili sesuatu yang nyata seperti komputer. Setiap *object* dalam sistem memiliki tiga karakteristik, yaitu *State* (stataus), *Behaviour* (sifat) dan *Identity* (identitas).

Class adalah deskripsi sekelompok *objec*t dari *property* (atribut), sifat (operasi) relasi antar *object* dan sematik yang umum. *Class* merupakan *template* untuk membentuk *objex*t. Setiap *objec*t merupakan contoh dari beberapa *class* dan *object* tidak dapt menjadi contoh lebih dari *satu class*. Penamaan *class* menggunakan kata benda tunggal yang merupakan abtraksi yang terbaik. Pada UML , *class* digambarkan dengan segi empat yang dibagi. Bagian atas merupakan nama dari *class*. Bagian yang tengah merupakan struktur dari *class* (atribut) dan bagian bawah merupakan sifat dari *class* (operasi).

2.9.2 Interaction Diagram

1. Use Case Relization

Fungsionalitas *use case* direpresentasaikan dengan aliran peristiwa–peristiwa. Skenario digunakan untuk menggambarkan bagaimana u*se case–use case* direalisasikan sebagai interaksi antara *object–object. Use case realization* menggambarkan bagaimana realisasi dari setiap *use case* yang ada pada *use case model*. Untuk menggambarkan bagaimana realasasi dari suatu *use case* dapat menggunakan beberapa diagram, diantaranya adalah *Class Diagram owned by Use Case Realization* serta *Interaction Diagram.*

2. Interaction Diagram

Menurut Hans-Erik merupakan model yang menjelaskan bagaimana sejumlah *object* bekerja sama dalam beberapa kelakuan. *Interaction Diagram* menerangkan kelakuan dari suatu *use case.* Diagram ini menggambarkan sejumlah *object* dari pesan yang dijalankan antar *object* dengan *use case.* Ketika kita memberikan pesan, aksi yang dihasilkan adalah sebuah pernyataan tereksekusi yang membentuk abstraksi dari prosedur komputasi. Dalam UML, kita dapat memodelkan beberapa jenis aksi , yaitu :

1. Call : memanggil operasi yang ada pada *object, object* mungkin mengirim ke dirinya sendiri , menghasilkan pemanggilan lokal dari operasi
2. Return : mengembalikan nilai dari caller
3. Send : mengirimkan sinyal ke object
4. Create : membuat sebuah object
5. Destroy : mematikan *object, object* mungkin saja mematikan diri.

Interaction diagram digunakan ketika kita ingin melihat kelakuan dari beberapa *object* dalam *use case* tunggal. Diagram ini baik saat menunjukan kolaborasi diantara *object-object*, namun kurang baik dalam mendefinisikan *behavior*. Ada dua macam *Interaction Diagram* yaitu : *Sequence Diagra*m dan *Collaboration Diagram.*

1. Sequence Diagram

Menurut Hans-Erik sequence Diagram menggambarkan interaksi antara sejumlah *object* dalam urutan waktu. Kegunaannya untuk menunjukan rangkaian pesan yang

dikirim antar *object* juga interaksi antar *object* yang terjadi pada titik tertentu dalam ekseskusi sistem. Dalam UML, *object* pada *diagram sequence* digambarkan dengan segi empat yang berisi nama dari *object* yang digaris bawahi. Pada *object* terdapat 3 cara untuk menamainya yaitu : *object* , nama *object* dan *class* serta nama *class.* Dalam *diagram sequence* setiap *object* hanya memiliki garis yang digambarkan garis putus – putus ke bawah. Pesan antar *object* digambarkan dengan anak panah dari *object* yang mengirimkan pesan ke *object* yang menerima pesan.

2. Collaboration Diagram

Collaboration Diagram merupakan cara alterantif untuk menggambarkan skenario dari sistem. Diagram ini menggambarkan interaksi *object* yang diatur *object* sekelilingnya dan hubungan antara setiap *object* dengan object yang lainnya.

Collaboration diagram berisi :

- *Objec*t yang digambarkan dengan segiempat
- Hubungan antara *object* yang digambarkan dengan garis penghubung
- Pesan yang digambarkan dengan teks danb panah dari object yang mengirim pesan ke penerima pesan

3. Perbedaan Sequence Diagram dengan Collaboration Diagram

Sequence Diagram memberikan cara untuk melihat skenario dari sistem berdasarkan waktu (apa yang terjadi pertama kali , apa yang terjadi selanjutnya). User akan lebih mudah membaca dan mengerti tipe diagram ini. Karenanya, sangat berguna pada fase analisis awal. Sedangkan *Collaboration Diagram* cenderung untuk memberikan gambaran besar dari skenario selama kolaborasi

disusun dari object sekellilingnya dan hubungan antar *object* yang satu dengan lainnya. Diagram ini akan nampak digunakan pada pengembangan tahap desain, ketika kita merancang implemantasi dari hitungan.

2.10 Activity Diagram

Activity Diagram memodelkan *work flow* proses bisnis dan urutan akitfitas dalam sebuah proses. Diagram ini sangat mirip dengan *flowchart* karena memodelkan *workflow* dari satu aktivitas ke aktivitas lainnya atau dari aktivitas ke status.

Elemen – elemen activity diagram :

1. *Status start*(Mulai) dan *end* (akhir)
2. Aktifitas yang merepresentasikan sebuah langkah dalam *workflow*
3. *Transition* menunjukan terjadinya perubahan status aktifitas *(Transition show what sate follows another)*
4. Keputusan yang menunjukan alternatif dalam *workflow*..
5. *Synchronization bars* yang meunjukan subflow parallel. *Syncronization bars* dapat digunakan untuk menunjukan *concurent thread* pada *workflow* proses bisnis
6. *Swimlanes* yang merepresentasikan *role* bisnis yang bertanggung jawab pada aktifitas yang berjalan.

2.11 Pengenalan *Artificial Intellegence* (Kecerdasan Buatan)

Kecerdasan Buatan (bahasa Inggris: *Artificial Intelligence* atau *AI*) didefinisikan sebagai kecerdasan yang ditunjukkan oleh suatu *entitas* buatan. Sistem seperti ini umumnya dianggap komputer. Kecerdasan diciptakan dan dimasukkan ke dalam suatu mesin (komputer) agar dapat melakukan pekerjaan seperti yang dapat dilakukan manusia. Beberapa macam bidang yang menggunakan kecerdasan buatan antara lain sistem pakar, permainan komputer (*games*), *logika fuzzy*, jaringan syaraf tiruan dan robotika.

Banyak hal yang kelihatannya sulit untuk kecerdasan manusia, tetapi untuk Informatika relatif tidak bermasalah. Seperti contoh: mentransformasikan persamaan, menyelesaikan persamaan integral, membuat permainan catur atau Backgammon. Di sisi lain, hal yang bagi manusia kelihatannya menuntut sedikit kecerdasan, sampai sekarang masih sulit untuk direalisasikan dalam Informatika. Seperti contoh: Pengenalan Obyek/Muka, bermain Sepakbola.

Walaupun AI memiliki konotasi fiksi ilmiah yang kuat, AI membentuk cabang yang sangat penting pada ilmu komputer, berhubungan dengan perilaku, pembelajaran dan adaptasi yang cerdas dalam sebuah mesin. Penelitian dalam AI menyangkut pembuatan mesin untuk mengotomatisasikan tugas-tugas yang membutuhkan perilaku cerdas. Termasuk contohnya adalah pengendalian, perencanaan dan penjadwalan, kemampuan untuk menjawab diagnosa dan pertanyaan pelanggan, serta pengenalan tulisan tangan, suara dan wajah. Hal-hal seperti itu telah menjadi disiplin ilmu tersendiri, yang memusatkan perhatian pada penyediaan solusi masalah kehidupan yang nyata. Sistem AI sekarang ini sering digunakan dalam bidang ekonomi, obat-obatan, teknik dan militer, seperti yang

telah dibangun dalam beberapa aplikasi perangkat lunak komputer rumah dan video game.

2.11.1 Dasar Pemikiran Kecerdasan Buatan

Secara garis besar, AI terbagi ke dalam dua faham pemikiran yaitu AI Konvensional dan Kecerdasan Komputasional (CI, Computational Intelligence). AI konvensional kebanyakan melibatkan metoda-metoda yang sekarang diklasifiksikan sebagai pembelajaran mesin, yang ditandai dengan formalisme dan analisis statistik. Dikenal juga sebagai AI simbolis, AI logis, AI murni dan AI cara lama (GOFAI, Good Old Fashioned Artificial Intelligence). Metoda-metodanya meliputi:

1) Sistem pakar: menerapkan kapabilitas pertimbangan untuk mencapai kesimpulan. Sebuah sistem pakar dapat memproses sejumlah besar informasi yang diketahui dan menyediakan kesimpulan-kesimpulan berdasarkan pada informasi-informasi tersebut.
2) Petimbangan berdasar kasus
3) Jaringan Bayesian
4) AI berdasarkan tingkah laku: metoda modular pada pembentukan sistem AI secara manual

Kecerdasan komputasional melibatkan pengembangan atau pembelajaran iteratif (misalnya penalaan parameter seperti dalam sistem koneksionis. Pembelajaran ini berdasarkan pada data empiris dan diasosiasikan dengan AI non-simbolis, AI yang tak teratur dan perhitungan lunak. Metoda-metoda pokoknya meliputi:

1) Jaringan Syaraf: sistem dengan kemampuan pengenalan pola yang sangat kuat

2) Sistem Fuzzy: teknik-teknik untuk pertimbangan di bawah ketidakpastian, telah digunakan secara meluas dalam industri modern dan sistem kendali produk konsumen.
3) Komputasi Evolusioner: menerapkan konsep-konsep yang terinspirasi secara biologis seperti populasi, mutasi dan "survival of the fittest" untuk menghasilkan pemecahan masalah yang lebih baik.

Metoda-metoda ini terutama dibagi menjadi algoritma evolusioner (misalnya algoritma genetik) dan kecerdasan berkelompok (misalnya algoritma semut)

Dengan sistem cerdas hibrid, percobaan-percobaan dibuat untuk menggabungkan kedua kelompok ini. Aturan inferensi pakar dapat dibangkitkan melalui jaringan syaraf atau aturan produksi dari pembelajaran statistik seperti dalam ACT-R. Sebuah pendekatan baru yang menjanjikan disebutkan bahwa penguatan kecerdasan mencoba untuk mencapai kecerdasan buatan dalam proses pengembangan evolusioner sebagai efek samping dari penguatan kecerdasan manusia melalui teknologi.

2.11.2 Sejarah *Artificial Intellegent*

Menurut René Descartes pada awal abad 17, mengemukakan bahwa tubuh hewan bukanlah apa-apa melainkan hanya mesin-mesin yang rumit. Blaise Pascal menciptakan mesin penghitung digital mekanis pertama pada 1642. Pada 19, Charles Babbage dan Ada Lovelace bekerja pada mesin penghitung mekanis yang dapat diprogram.

Bertrand Russell dan Alfred North Whitehead menerbitkan Principia Mathematica, yang merombak logika formal. Warren McCulloch dan Walter Pitts

menerbitkan "Kalkulus Logis Gagasan yang tetap ada dalam Aktivitas " pada 1943 yang meletakkan pondasi untuk jaringan syaraf.

Tahun 1950-an adalah periode usaha aktif dalam AI. Program AI pertama yang bekerja ditulis pada 1951 untuk menjalankan mesin Ferranti Mark I di University of Manchester (UK): sebuah program permainan naskah yang ditulis oleh Christopher Strachey dan program permainan catur yang ditulis oleh Dietrich Prinz. John McCarthy membuat istilah "kecerdasan buatan " pada konferensi pertama yang disediakan untuk pokok persoalan ini, pada 1956. Dia juga menemukan bahasa pemrograman Lisp. Alan Turing memperkenalkan "Turing test" sebagai sebuah cara untuk mengoperasionalkan test perilaku cerdas. Joseph Weizenbaum membangun ELIZA, sebuah chatterbot yang menerapkan psikoterapi Rogerian.

Selama tahun 1960-an dan 1970-an, Joel Moses mendemonstrasikan kekuatan pertimbangan simbolis untuk mengintegrasikan masalah di dalam program Macsyma, program berbasis pengetahuan yang sukses pertama kali dalam bidang matematika. Marvin Minsky dan Seymour Papert menerbitkan Perceptrons, yang mendemostrasikan batas jaringan syaraf sederhana dan Alain Colmerauer mengembangkan bahasa komputer Prolog. Ted Shortliffe mendemonstrasikan kekuatan sistem berbasis aturan untuk representasi pengetahuan dan inferensi dalam diagnosa dan terapi medis yang kadangkala disebut sebagai sistem pakar pertama. Hans Moravec mengembangkan kendaraan terkendali komputer pertama untuk mengatasi jalan berintang yang kusut secara mandiri.

Pada tahun 1980-an, jaringan syaraf digunakan secara meluas dengan algoritma perambatan balik, pertama kali diterangkan oleh Paul John Werbos pada 1974.

Tahun 1990-an ditandai perolehan besar dalam berbagai bidang AI dan demonstrasi berbagai macam aplikasi. Lebih khusus Deep Blue, sebuah komputer permainan catur, mengalahkan Garry Kasparov dalam sebuah pertandingan 6 game yang terkenal pada tahun 1997. DARPA menyatakan bahwa biaya yang disimpan melalui penerapan metode AI untuk unit penjadwalan dalam Perang Teluk pertama telah mengganti seluruh investasi dalam penelitian AI sejak tahun 1950 pada pemerintah AS.

2.12 Pengenalan Sistem Pakar

Menurut René Descartes sistem pakar adalah salah satu sub bidang *artificial intelligence* yang mendampingi Natural Language Processing (NLP) atau pengolahan bahasa alami, *robotics* atau robotika. Secara umum pengetahuan di dalam sistem pakar disimpulkan dari para pakar pada bidang yang spesifik itu, sehingga sistem pakar cenderung menjadi seorang spesialis yang hanya terfokus pada sejumlah masalah yang sempit dan terbatas. Seperti juga manusia, pengetahuan dalam sistem pakar diperoleh secara teori dan praktek. Sistem pakar yang cenderung menjadi seorang spesialis ini memiliki keuntungan sebagai berikut:

a. Membuat seorang yang awam pada bidang tertentu dapat bekerja seperti layaknya seorang pakar pada bidang tersebut.
b. Memudahkan pengoperasian peralatan yang kompleks karena sistem pakar dapat melatih pekerja yang tidak berpengalaman.
c. Dapat diandalkan. Sistem pakar tidak pernah lelah atau bosan, serta konsisten dan perhatian penuh dalam memberi jawaban.
d. Dapat bekerja dengan informasi yang tidak lengkap atau tidak pasti.

e. Meningkatkan output dan produktivitas. Sistem pakar dapat bekerja lebih cepat dari manusia. Keuntungan ini berarti mengurangi jumlah pekerja yang dibutuhkan sehingga dapat mereduksi biaya.
f. Meningkatkan kualitas. sistem pakar menyediakan nasehat yang konsisten serta mengurangi tingkat terjadinya kesalahan.
g. Kemampuan untuk memecahkan suatu masalah yang kompleks.
h. Merupakan arsip yang terpercaya dari sebuah bidang, sehingga user seolah-olah sedang berkonsultasi langsung dengan sang pakar, meskipun mungkin sang pakar sudah pensiun.
i. Pemindahan pengetahuan ke lokasi yang jauh dan memperluas jangkauan seorang pakar, yang dapat diperoleh dan dipakai dimana saja.

Melalui sistem pakar, sistem melakukan ekstrasi informasi tambahan dari user dengan memberikan sejumlah pertanyaan yang terkait dengan permasalahan selama berkonsultasi. Sistem pakar biasanya digunakan oleh :

a. Orang awam untuk meningkatkan kemampuan mereka dalam memecahkan masalah.
b. Seorang asisten yang berkemampuan seperti seorang pakar.

Dari poin-poin diatas tersebut, maka dapat disimpulkan bahwa sistem pakar pada umumnya memiliki ciri-ciri sebagai berikut :

a. Hanya terbatas pada bidang yang spesifik.
b. Dapat memberikan penalaran untuk data-data yang tidak lengkap atau tidak pasti.
c. Mengemukakan rangkaian alasan yang diberikannya dengan cara yang dapat dipahami.
d. Berdasarkan pada aturan tertentu.

e. Dirancang untuk dapat dikembangkan secara bertahap dan berkelanjutan.
f. Hasilnya atau outputnya bersifat nasehat atau anjuran.
g. Output tergantung pada dari komunikasi melalui dialog dengan user.
h. *Knowledge base* dan *inference engine* terpisah.

Umumnya sistem pakar diharapkan mampu untuk memenuhi fungsi-fungsi sebagai berikut :

a. Penyelesaian masalah, kemampuan untuk menyelesaikan masalah dengan tepat pada domain yang spesifik dengan informasi yang terbatas dan diperlukan (tidak harus lengkap).
b. Berinteraksi dengan user, sistem pakar harus dapat berinteraksi dengan user, dan menjelaskan suatu sistem dan kesimpulan-kesimpulan selama dan setelah proses penyelesaian masalah.

Dengan fungsi-fungsi sistem pakar tersebut, diharapkan sistem pakar dapat menggantikan keahlian dari seorang pakar pada domain yang sama. Karena pada kenyataannya, kemampuan kerja seorang ahlipun dipengaruhi oleh beberapa hal (misalnya perasaan pakar tersebut, kondisi pakar tersebut labil atau tidak, tingkat emosi pakar, dan masalah-masalah yang sedang dihadapi oleh sistem pakar), berbeda dengan sistem pakar yang bekerja pada kemampuan kerja yang terus konsisten, dimana pada beberapa kasus dapat diselesaikan lebih cepat oleh sistem pakar daripada oleh pakar itu sendiri.

Hal ini bukan berarti bahwa sistem pakar lebih unggul dibandingkan dengan pakar itu sendiri. Hal ini dapat disebabkan karena pada sistem pakar ada kemungkinan salah menemukan kesimpulan yang dihasilkan meskipun data-data yang diberikan adalah valid.

2.13 Ciri-ciri sistem pakar

Menurut (Shelly,1990; Setiawan 1993; Margianti,1995). Ciri-ciri sistem pakar adalah sebagai berikut:

1. Terbatas pada domain keahlian tertentu.
2. Dapat memberikan penalaran untuk data data yang tidak pasti.
3. Dapat mengemukan rangkaian alasan-alasan yang diberikannya dengan cara yang dapat dipahami.
4. Berdasarkan pada kaidah/Rule tertentu.
5. Dirancang untuk dapat dikembangkan secara bertahap.
6. Keluaranya bersifat anjuran.

Komponen sistem pakar terbagi menjadi empat bagian, yaitu:

1. Knowledge Base (Basis Pengetahuan)

   Knowledge Base merupakan inti dari program sistem pakar karena basis pengetahuan itu merupakan presentasi pengetahuan atau knowledge representation basis pengetahuan adalah sebuah basis data yang menyimpan aturan-aturan tentang suatu domain knowledge/pengetahuan tertentu. Basis pengetahuan ini terdiri dari kumpulan objek beserta aturan dan atributnya (sifat atau cirinya). Contoh : If hewan merupakan sayap dan bertelur then hewan jenis burung.

2. Working Memory (Basis Data atau Memori Kerja)

   Working memory adalah bagian yang mengandung semua fakta-fakta baik fakta awal pada saat sistem beroperasi maupun fakta-fakta pada saat pengambilan kesimpulan sedang dilaksanakan selama sistem pakar beroperasi basis data berada di adalam memori kerja.

3. Inference Engine (Mesin Inferensia)

   Inference Engine adalah bagian yang menyediakan mekanisme fungsi berfikir dan pola-pola penalaran sistem yang digunakan oleh seorang pakar.

   - Mekanisme ini akan menganalisa masalah tertentu dan selanjutnya akan mencari jawaban atau kesimpulan yang terbaik.
   - Mesin ini akan dimulai pelacakannya dengan mencocokan kaidah-kaidah dalam basis pengetahuan dengan fakta-fakta yang ada dalam basis data.

   Dua teknik Inference, yaitu:

   a. *Backward Chaining* (Pelacakan kebelakang)

   Melalui penalaranya dari sekumpulan hipotesis menuju fakta-fakta yang mendukung tersebut,jadi proses pelacakan berjalan mundur dimulai dengan menentukan kesimpulan yang akan dicari baru kemudian fakta-fakta pembangun kesimpulan atau *a Goal Driven*.

   b.*Forward Chaining* (Pelacakan ke depan)

   Forward Chaining merupakan kebalikan dari *Backward Chaining* yaitu mulai dari kumpulan data menuju kesimpulan. Suatu kasus kesimpulannya dibangun berdasarkan fakta-fakta yang telah diketahui atau data driven.

4. User Interface (Antarmuka Pemakai)

Antarmuka pemakai adalah bagian penghubung antara program sistem pakar dengan pemakai. Pada bagian memungkinkan pengguna untuk memasukkan instruksi dan informasi ke dalam sistem pakar serta menerima penjelasan dan kesimpulan

2.14 Sepeda Motor

Menurut Toto Suwarto sejalan dengan perkembangan dan kemajuan teknologi, sepeda motor merupakan salah satu jenis alat transportasi yang murah, efisien, dan antimacet, dan bisa juga digunakan untuk menjangkau daerah yang sulit dijangkau oleh transportasi lain seperti mobil. Secara umum, sepeda motor dapat berfungsi sebagai alat transportasi, baik digunakan secara pribadi, sebagai jasa layanan pos (pengantar surat), patroli motor (kepolisian), maupun sebagai sarana angkutan murah bagi masyarakat yaitu jasa ojek. Pertumbuhan jumlah sepeda motor sangat pesat seiring dengan tingkat ekonomi dan kebutuhan masyarakat terhadap alat transportasi yang murah dan terjangkau golongan ekonomi menengah ke bawah. Akan tetapi, seiring makin banyaknya jumlah sepeda motor dijalanan, ternyata menimbulkan masalah baru, yaitu pencemaran lingkungan berupa polusi udara. Knalpot sepeda motor asap dan menambah parahnya polusi udara di kota-kota besar. Sepeda motor '2 tak ' sering di jadikan 'kambing hitam' penyebab polusi udara. Mungkin, pernyataan ini tidak terlalu keliru sebab sebagaian besar motor 2-tak adalah mengeluarkan banyak asap. Asap kendaraan 2-tak mengandung karbon monoksida (CO), hidro karbon (HC), dan NO_x yang berbahaya bagi kesehatan manusia. Sementara itu sepeda motor 4-tak di kenal lebih ramah lingkungan karena jenis ini tidak mengeluarkan asap, kecuali terjadi kerusakan pada system pembakarannya. Sepeda motor 4-tak juga lebih irit bahan bakar dibandingkan sepeda motor 2-tak. Biaya yang harus dikeluarkan oleh pengguna sepeda motor 2-tak jauh lebih besar dibandingkan pengguna sepeda motor 4-tak. Sealin itu, pengguna sepeda motor 2-tak juga perlu menyiapkan biaya tambahan untuk membeli oili samping. Dengan pertimbangan hemat biaya dan rendah polusi masyarakat modern cenderung lebih memilih

sepeda motor 4-tak sebagai kendaraan pribadinya. Namun, sepeda motor 4-tak membutuhkan perawatan yang lebih itensif dibandingkan sepeda moto 2-tak. Hal ini disebabkan sepeda motor 4-tak mempunyai susunan mesin yang lebih rumit. Kepala silindernya tersusun atas klep-klep untuk mengatur pemasukan dan pembuangan bahan bakar.

2.15 Komponen – Komponen Sepeda Motor

Menurut Drs.M.Suratman komponen sepeda motor, baik dua tak maupun empat tak, terdiri dari :

1. Spatbor depan
2. Lampu besar (dim)
3. Lampu sein
4. Speedometer
5. Kunci kontak
6. Kaca spion
7. Tuas kopling
8. Tutup tangki
9. Tangki bensin
10. Keran bensin
11. Lampu belakang (rem)
12. Lampu sen belakang
13. Spatbor belakang

14. Sockbreaker belakang

15. Sproeket dan rantai

16. Penyangga tengah

17. Pedal perseneling

18. Generator

19. Klakson

20. Brake caliper

21. Brake disk

22. Kotak sikring

23. Kotak batere

24. Tuas rem depan

25. Master cylinder

26. Sockbreaker depan

27. Pipa gas buang

28. Motor stater

29. Blok mesin (engine)

30. Tuas rem belakang

31. Pedal star (kick starter)

32. Knalpot

33. Tuas cam rem belakang

2.16 Jenis – Jenis Kerusakan

Menurut Drs.M.Suratman jenis-jenis kerusakan yang dibahas pada sistem informasi ini meliputi kerusakan-kerusakan yang sering terjadi,.dalam sistem informasi ini dibahas mengenai kemungkinan adanya kerusakan.

Proses pendeteksian kerusakan dari sistem informasi ini di ambil dari buku SERVIS DAN TEKNIK REPARASI SEPEDA MOTOR yang dapat di ikuti oleh orang yang belum mengetahui kerusakan pada sepeda motor, serta untuk mengetahui seberapa parahnya kerusakan sepeda motor yang terjadi.

Semua sistem informasi yang ada pada dasarnya, menjelaskan kerusakan-kerusakan utama yang ada pada sepeda motor meliputi : kelistrikan, mesin, dan rangka.

2.17 Microsoft Visual Basic

Visual Basic adalah sebuah bahasa pemrograman generasi ke-4 (*fourth generation programming language*) yang diciptakan oleh Microsoft. Bill Gates, pendiri Microsoft, memulai bisnis softwarenya dengan mengembangkan Basic Interpreter untuk IBM. Perkembangan berikutnya ialah diluncurkannya BASICA (basic-advanced) untuk DOS. Setelah BASICA, Microsoft meluncurkan Microsoft QuickBasic dan Microsoft Basic (dikenal juga sebagai Basic Compiler). Sejarah BASIC di tangan Microsoft sebagai bahasa yang diinterpretasi (BASICA) dan juga bahasa yang dikompilasi (BASCOM) membuat *Visual Basic* diimplementasikan sebagai gabungan keduanya. Programmer yang menggunakan *Visual Basic* bisa memilih kode terkompilasi atau kode yang harus diinterpretasi sebagai hasil executable dari kode VB. Sayangnya, meskipun sudah terkompilasi jadi bahasa mesin, DLL bernama MSVBVMxx.DLL tetap dibutuhkan.

Visual Basic merupakan bahasa yang mendukung OOP, namun tidak sepenuhnya. Beberapa karakteristik obyek tidak dapat dilakukan pada Visual Basic, seperti *Inheritance* tidak dapat dilakukan pada class module. *Polymorphism* secara terbatas bisa dilakukan dengan mendeklarasikan *class module* yang memiliki *Interface* tertentu.

Visual Basic menjadi populer karena kemudahan desain form secara visual dan adanya kemampuan untuk menggunakan komponen-komponen *ActiveX* yang dibuat oleh pihak lain. Namun komponen *ActiveX* memiliki masalahnya tersendiri yang dikenal sebagai *DLL hell*. Pada Visual Basic.NET, Microsoft mencoba mengatasi masalah *DLL hell* dengan mengubah cara penggunaan komponen (menjadi independen terhadap registry).

2.18 Uji Kepuasan

- Defisnisi

Pengujian adalah pengkajian atas tanggapan konsumen dan penyaluran terhadap masalah-masalah perlakuan dan penggunaannya.

Kepuasan Adalah perasaan senang atau kecewa seseorang yang muncul setelah membandingkan antara persepsi/kesan terhadap kinerja dan harapannya.

Uji kepuasaan adalah pengkajian atas tanggapan responden dan penyaluran masalah-masalah perlakuan, penggunaan atas kinerja (atau hasil) dan harapan jika kinerja berada dibawah harapan maka user tidak puas, jika kinerja memenuhi harapan, user puas, jika kinerja melebihi harapan maka user amat puas.

- Alasan Pemakaian

Dikarenakan metode analisis deskriftif yang di kemukakan oleh Umar Husein dapat dipahami dengan mudah serta sangat efektif dan sesuai dengan dengan klasifikasi system yang penulis buat