BAB II
LANDASAN TEORI

2.1 Sistem Pakar

Aplikasi yang mungkin berkembang saat ini yaitu sistem pakar (*expert system*). Pada dasarnya *expert sistem* terdiri atas banyak *database* dan seperangkat aturan yang dapat mencari sendiri dari database tersebut menjadi solusi terbaik atas suatu masalah. Sistem pakar diambil dari istilah *knowledge base expert system*. *knowledge base expert system* dibentuk dari *knowledge base system* yang merupakan hasil dari proses *knowledge engineering*.

Sistem pakar (*expert sistem*) adalah sistem yang berusaha mengadopsi pengetahuan manusia ke komputer, agar komputer dapat menyelesaikan masalah seperti yang biasa dilakuakan oleh para ahli. Sistem pakar yang baik dirancang agar dapat menyelesaikan suatu permasalahan tertentu dengan meniru kerja dari para ahli. (Kusumadewi, 2003:109).

Ciri-ciri sistem pakar 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.

2.1.1 Keuntungan Sistem Pakar

Manfaat yang dapat diambil dengan adanya sistem pakar antara lain sebagai berikut:

1) Memungkinkan orang awam bisa mengerjakan pekerjaan para ahli.
2) Bisa melakukan proses secara berulang secara otomatis.

3) Menyimpan pengetahuan dan keahlian para pakar.
4) Meningkatkan output dan produktivitas.
5) Meningkatkan kualitas.
6) Mampu mengambil dan melestarikan keahlian para pakar
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 dan mengandung ketidakpastian
12) Sebagai media pelengkap dalam penelitian.
13) Meningkatkan kapabilitas dalam penyelesaian masalah
14) Menghemat waktu dalam pengambilan keputusan.

2.1.2 Kelemahan sistem pakar

Sistem pakar memiliki beberapa kelemahan, antara lain sebagai berikut:

1) Biaya yang diperlukan untuk membuat dan memeliharanya sangat mahal dan sulit dikembangkan. Hal ini tentu saja erat kaitannya dengan ketersediaan pakar di bidangnya.
2) Sistem Pakar tidak 100% bernilai benar.

2.1.3 Konsep Dasar Sistem Pakar

Konsep-konsep dasar yang terdapat dalam sebuah sistem pakar antara lain sebagai berikut:

1) Keahlian

   Kelebihan penguasaan pengetahuan di bidang tertentu yang diperoleh dari pelatihan, membaca atau pengalaman.
2) Seorang ahli

   Seorang ahli adalah seseorang yang mampu menjelaskan suatu tanggapan, mempelajari hal-hal baru seputar topik permasalahan *(domain),* menyusun kembali pengetahuan jika dipandang perlu,

memecahkan aturan-aturan jika dibutuhkan, dan menentukan relevan tidaknya keahlian mereka.

3) Pengalihan keahlian

Pengalihan dari para ahli ke komputer untuk kemudian dialihkan lagi ke orang lain yang bukan ahli, merupakan tujuan utama dari sistem pakar. Proses ini membutuhkan 4 aktivitas yaitu: tambahan pengetahuan (dari para ahli atau sumber-sumber lainnya), representasi pengetahuan (ke komputer), inferensi pengetahuan, dan pengalihan pengetahuan ke *user*.

4) Inferensi

Salah satu fitur yang harus dimiliki oleh sistem pakar adalah kemampuan untuk menalar. Jika keahlian-keahlian sudah tersimpan sebagai basis pengetahuan *(knowledge base)* maka komputer harus dapat membuat inferensi. Proses inferensi ini dikemas dalam bentuk motor inferensi *(Inference Engine).*

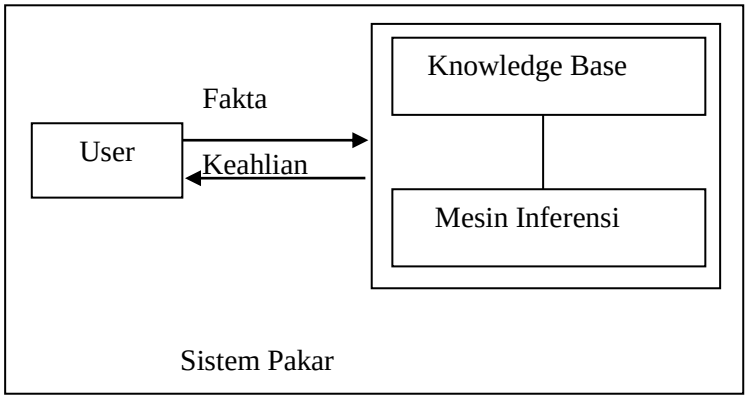
5) Aturan

Sebagian besar sistem pakar dalam bentuk *rule based system*, yaitu pengetahuan disimpan dalam bentuk aturan-aturan.

6) Kemampuan

Merupakan kemampuan untuk merekomendasi.

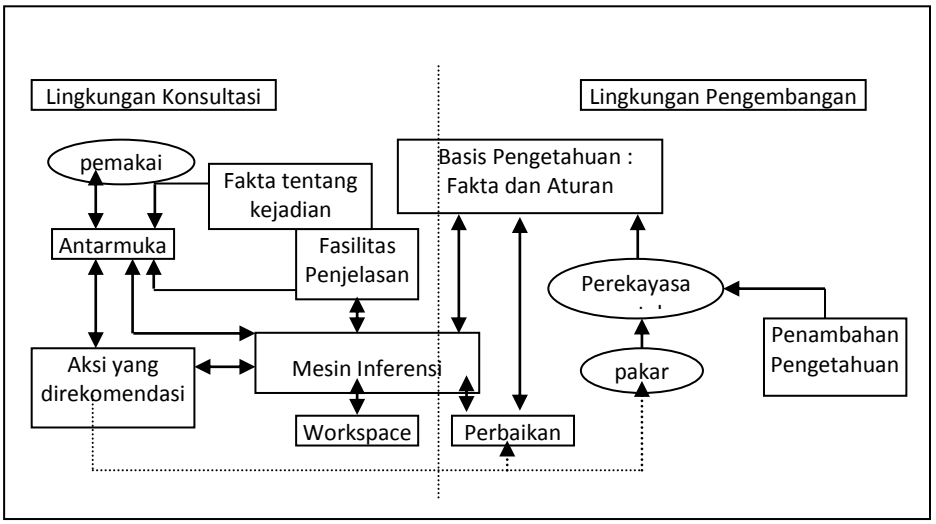
Gambar 2.1 berikut ini menggambarkan konsep dasar sistem pakar. Pengguna menyampaikan fakta atau informasi untuk sistem pakar dan kemudian menerima saran dari pakar atau jawaban ahlinya.



Gambar 2.1 Konsep Dasar Sistem Pakar

2.1.4 Struktur Sistem Pakar (*Expert System*)

Sistem pakar terdiri dari 2 bagian pokok, yaitu lingkungan pengembangan (*development environment*) dan lingkungan konsultasi (*consultation environment*). Lingkungan pengembangan digunakan sebagai pembangun sistem pakar baik dari segi pembangun komponen maupun basis pengetahuan. Lingkungan konsultasi digunakan oleh seseorang yang bukan ahli untuk berkonsultasi. Gambar 2.2 berikut ini merupakan gambar struktur sistem pakar. (Kusumadewi, 2003:113-115).



Gambar 2.2 Struktur Sistem Pakar

2.1.5 Basis Pengetahuan (*Knowledge Base*)

Basis pengetahuan berisi pengetahuan-pengetahuan dalam penyelesaian masalah, tentu saja didalam domain tertentu. Ada 2 bentuk pendekatan basis pengetahuan yang umum digunakan, yaitu:

a) Penalaran berbasis aturan (*Rule-Based Reasoning*)

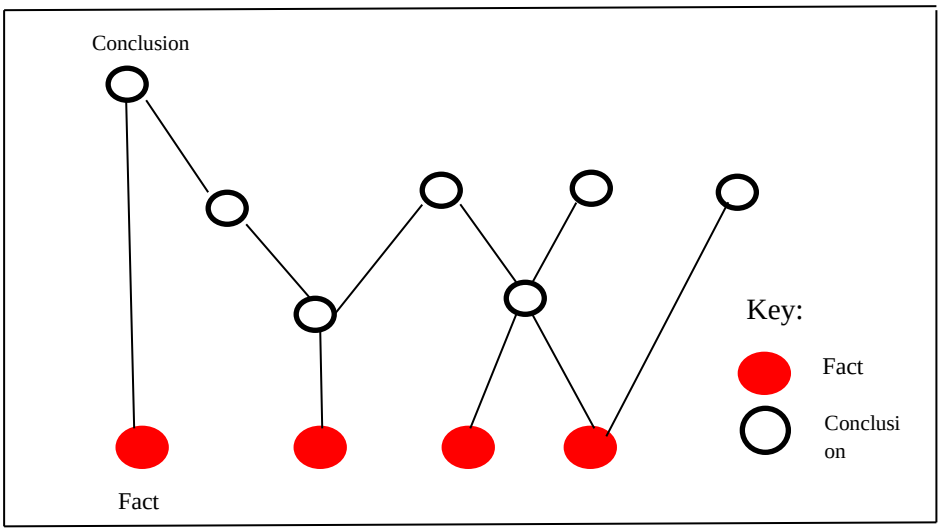
   Pada penalaran berbasis aturan, pengetahuan direpresentasikan dengan menggunakan aturan berbentuk *IF-THEN*. Bentuk ini digunakan apabila kita memiliki sejumlah pengetahuan pakar pada suatu permasalahan tertentu, dan si pakar dapat menyelesaikan masalah tersebut secara berurutan. Disamping itu, bentuk ini juga digunakan apabila dibutuhkan penjelasan tentang jejak langkah-langkah untuk pencapaian solusi.

b) Penalaran berbasis kasus (*Case-Based Reasoning*)

Pada penalaran berbasis kasus, basis pengetahuan akan berisi solusi-solusi yang telah dicapai sebelumnya, kemudian akan diturunkan suatu solusi untuk keadaan yang terjadi sekarang (fakta yang ada). Bentuk ini digunakan apabila *user* menginginkan untuk tahu lebih banyak lagi pada kasus-kasus yang hamper sama (mirip). Selain itu, bentuk ini juga digunakan apabila kita telah memiliki sejumlah situasi atau kasus tertentu dalam basis pengetahuan. (Kusumadewi, 2003:115-116).

2.1.6 Metode *Forward Chaining*

Sutojo, Mulyanto dan Suhartono (2011:171) mendefinisikan "*forward chaining* adalah teknik pencarian yang dimulai dengan fakta yang diketahui kemudian mencocokan fakta-fakta tersebut dengan bagian *IF* dari *rules IF-THEN".* Bila ada fakta tersebut dengan bagian *IF*, maka *rule* tersebut dieksekusi. Bila sebuah *rule* dieksekusi, maka sebuah fakta baru (bagian *THEN*) ditambahkan kedalam *database*. Setiap kali pencocokan, dimulai dari *rule* teratas. Setiap rule hanya boleh dieksekusi sekali saja. Proses pencocokan akan berhenti bila tidak ada lagi *rule* yang bisa dieksekusi. Contoh A=B, B=C, C=D, D=E dan seterusnya. Untuk memahami cara kerja *forward chaining*, perhatikan gambar 2.3 berikut:



Gambar 2.3 Gambar Kerja *Forward Chaining/Bottom Up*

2.1.7 Mesin Inferensi

Mesin inferensi adalah program komputer yang memberikan metodologi untuk penalaran tentang informasi yang ada dalam basis pengetahuan dan dalam *workplace* dan untuk memformulasikan kesimpulan.

Terdapat dua pendekatan untuk mengontrol inferensi dalam sistem pakar berbasis aturan yaitu:

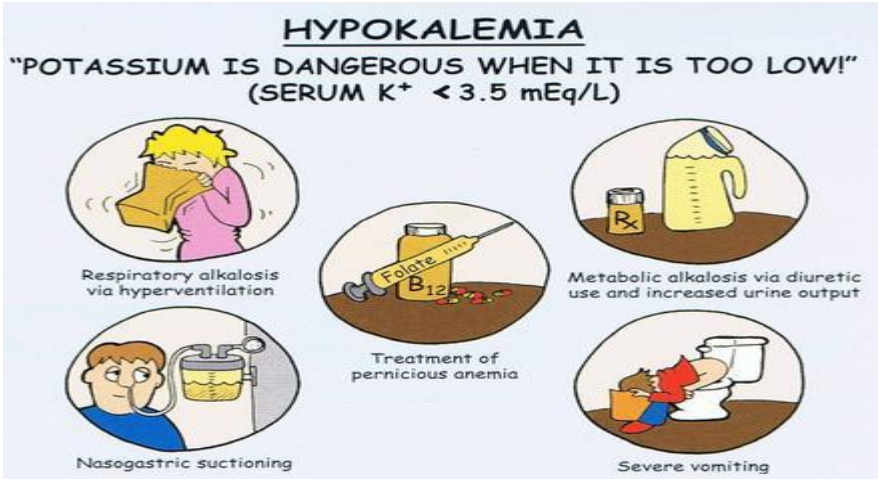
1) Pelacakan ke belakang *(backward chaining)* adalah proses yang dimulai dari tujuan. Jika informasi-informasi atau nilai dari atribut-atribut yang mengarah ke kesimpulan tersebut sesuai dengan data yang diberikan maka kesimpulan tersebut merupakan solusi yang dicari, jika tidak sesuai maka kesimpulan tersebut bukan merupakan solusi yang dicari. Sehingga strategi ini disebut juga *goal-driven.*
2) Pelacakan ke depan *(forward chaining)* adalah penalaran yang dimulai dari fakta terlebih dahulu untuk menguji hipotesis. Fakta-fakta tersebut dicari suatu kesimpulan yang menjadi solusi permasalahan yang selanjutnya runut maju memulai proses pencarian data, sehingga strategi ini disebut *data-driven*

Kedua metode inferensi tersebut dipengaruhi oleh tiga macam penelusuran, yaitu *Depth-first search*, *Breath-first search* dan *Best-first search.*

a) *Breath First search*, merupakan pencarian yang dilakukan dengan mengujungi tiap-tiap node secara sistematis pada setiap level hingga keadaan (*goal state*). (Desiani dan Arhami, 2006:27).
b) *Depth-first search*, merupakan pencarian yang dilakukan dengan mengujungi tiap-tiap node secara mendalam hingga yang paling akhir (*dead-end*). (Desiani dan Arhami, 2006:30).
c) *Best-first search*, bekerja berdasarkan kombinasi kedua metode sebelumnya. (Desiani dan Arhami, 2005:33)

2.2 Penyakit Hipokalemia

Jurnal penelitian sebelumnya oleh Matsuda (2010) menjelaskan, hipokalemia adalah suatu keadaan dimana konsentrasi kalium dalam darah kurang dari 3.5 mEq/L darah. Hipokalemia dapat terjadi karena adanya faktor tertentu, misalnya makanan dengan kadar karbohidrat tinggi, istirahat sesudah latihan fisik, perjalanan jauh, pemberian obat, operasi, menstruasi, konsumsi alkohol dan lain-lain. Jika konsentrasi kalium darah terlalu rendah, biasanya disebabkan oleh ginjal yang tidak berfungsi secara normal atau terlalu banyak kalium yang hilang melalui saluran pencernaan (karena diare, muntah, penggunaan obat pencahar dalam waktu yang lama atau polip usus besar). Gejala dan tanda penyakit hipokalemia dapat dilihat pada gambar 2.4 dibawah in:



Gambar 2.4 Gejala dan Tanda Penyakit Hipokalemia

Pencegahan penyakit hipokalemia antara lain:

1) Hindari minum es
2) Jangan makan karbohidrat terlalu banyak, karena karbohidrat sifatnya mengikat kalium
3) Batasi minum kopi atau minuman yang banyak mengandung gula
4) jika timbul gejala usahakan jangan dibawa tidur, karena bisa menyebabkan kelum puhan total
5) Usahakan banyak bergerak disaat awal gejala penyakit

Penyebab hipokalemia antara lain:

1) Gangguan fungsi ginjal
2) Terlalu banyak kalium yang hilang melalui saluran cerna, misalnya karena diare, mutah, pemakaian obat pencahar untuk waktu yang lama atau polip usus besar
3) Konsumsi sejumlah besar kayu manis atau produk tembakau tertentu
4) Gangguan tertentu, misalnya sindroma cushing, sindroma liddle, dan sindroma fanconi
5) Pemakaian obat tertentu, seperti diuretik, insulin, albuterol, dan teofilin.

2.2.1 Manfaat Kesehatan dari Kalium

Kalium melakukan sejumlah fungsi dalam tubuh, dan manfaat kesehatan dari kalium antara lain :

1) Mencegah terjadinya stroke di otak manusia
2) Memainkan peran kunci dalam fungsi otot jantung, dan juga dalam kontraksi otot
3) Mencegah timbulnya osteoporosis, penyakit yang ditandai dengan kepadatan mineral tulang berkurang, yang mengarah ke tulang keropos
4) Memainkan peran kunci dalam menjaga keseimbangan garam dan air dalam tubuh, bersama dengan natrium
5) Menjaga pH darah, yang merupakan ukuran dari keasaman
6) Asupan kalium dalam jumlah yang cukup dapat mencegah batu ginjal

2.2.2 Makanan Kaya Kalium

Asupan makanan harian yang direkomendasikan kalium adalah sekitar 3,8 gram. Sumber penting kalium meliputi daging, beberapa jenis ikan, buah-buahan, sayuran, dan kacang-kacangan. Selain itu, produk susu juga merupakan sumber yang kaya kalium. Sumber yang kaya kalium dapat dilihat pada tabel 2.1 sebagai berikut:

Tabel 2.1 Sumber kaya kalium

Sayuran	Buah-buahan
Wortel	**Aprikot**
Ubi Jalar	**Nektarin**
Seledri	**Alpukat**
Biji Kering	**Pisang**
Tomat	**Kismis**
Bayam	**Melon Honeydew**
Kentang	**Kurma**

2.2.3 Penyebab Umum dari Penyebab Hipokalemia

Penyebab Umum dari Penyebab Hipokalemia adalah sebagai berikut:

a. Obat – obatan
   1) Thiazide dan diuretik loop, aminoglikosida, amfoterisin B, β2-agonis dan steroid adrenal.
   2) Penyalahgunaan pencahar kronis
b. Gastrointestinal
   1) Muntah, yang mungkin disebabkan oleh diri sendiri pada pasien dengan beberapa gangguan makan, menyebabkan hilangnya asam klorida, alkalosis metabolik, tingkat rendah klorida urin, dan meningkatkan ekskresi kalium ginjal
   2) Obstruksi lambung, merupakan gangguan pasase dari isi usus akibat sumbatan yang bisa disebabkan oleh tukak lambung pada orang dewasa atau dengan stenosis pilorus pada bayi dan anak-anak, menghasilkan pola metabolik mirip dengan muntah
c. Renal
   1) Asidosis tubulus ginjal, baik proksimal (tipe 2) dan distal (tipe 1)

2) Kekurangan magnesium yang berhubungan dengan alkoholisme dapat menyebabkan hipokalemia refraktori
3) Penggunaan cisplatin, gentamisin, atau levodopa juga dapat menyebabkan hipokalemia refraktori.

2.2.4 Penyebab Lain dari Penyebab Hipokalemia

Penyebab lain yang umum terjadi dari penyebab hipokalemia ini antara lain sebagai berikut:

a) Penyalahgunaan alkohol menyebabkan hipokalemia melalui kombinasi pola makan yang buruk, muntah, dan / atau penipisan magnesium bersamaan
b) Asupan makanan yang buruk

2.2.5 Penyebab Serius dari Penyebab Hipokalemia

Penyebab serius dari penyebab hipokalemia antara lain sebagai berikut:

a) Ketoasidosis diabetik menyebabkan deplesi kalium, tetapi hipokalemia dapat bertopeng sampai insulin dan cairan yang diberikan, yang mengarah pada masuknya cepat kalium ke dalam sel
b) Terapi vitamin B12 untuk anemia megaloblastik menghasilkan eritrosit baru, sehingga masuknya intraseluler kalium, yang dapat menghasilkan hipokalemia parah dan berpotensi mematikan.

2.2.6 Faktor Penyebab dari Penyebab Hipokalemia

Faktor penyebab hipokalemia ini adalah sebagai berikut:

a) Gagal jantung kongestif: pengobatan diuretik yang dapat diperburuk oleh hiperaldosteronisme sekunder
b) Gangguan pencernaan: muntah, diare, hisap lambung
c) Diabetes, ketika tidak terkontrol

d) Gangguan makan: self-induced muntah, penyalahgunaan obat pencahar atau diuretik, diet sangat rendah kalori
e) Deplesi magnesium
f) Alkoholisme, yang mungkin berhubungan dengan asupan rendah makanan, muntah, dan magnesium deplesi
g) Penyakit Ginjal: ginjal tubular acidosis (tipe 1 dan 2), nefritis interstitial
h) Hipertensi: pengobatan diuretik, hiperaldosteronisme

2.3 Penelitian Terdahulu

Tabel 2.2 berikut merupakan penelitian terdahulu yang telah dilakukan terkait dengan metode *forward chaining.*

Tabel 2.2 Penelitian Terdahulu

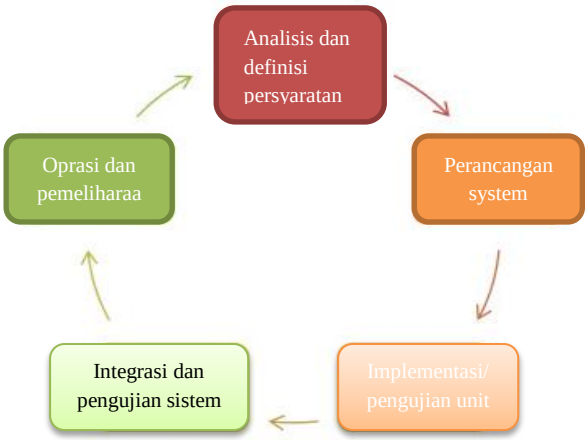
Nama	Judul	Terbit/Tahun	Keterangan
Rosa Delima	Penerapan *Forward Chaining* Pada Program Diagnosa Penyakit Anak Autisme	Jurnal Universitas Kristen Duta Wacana, Yogyakarta 2009	Autisme merupakan gangguan mental pada anak yang membuat anak sulit bersosialisasi. Sistem bertujuan untuk memberikan kemudahan pada orang tua untuk mengetahui autisme pada anak
Joko Purwadi	Program Bantu Diagnosa Gangguan Kesehatan Dengan Metode *Forward Chaining*	Jurnal Universitas Kristen Duta Wacana, Yogyakarta 2010	Sistem pakar untuk mendignosa penyakit kehamilan dirancang sebagai alat bantu untuk membuat para tenaga medis dan masyarakat dalam mendignosa penyakit kehamilan secara dini.

Tabel 2.2 (Lanjutan)

Anton Setiawan Hoggowibowo	Sistem Pakar Diagnosa Penyakit Tanaman Padi Berbasis Web Dengan *Forward Chaining*	Jurnal Sekolah Tinggi Teknologi Adisutijpto, Yogyakarta 2013	Sistem pakar dignosa penyakit tanaman padi dibuat dengan tujuan membantu pakar mengetahui jenis penyakit tanaman padi serta dapat memberikan informasi mengenai penyakit pada padi
Ida Bagus Dhany Satwika	Rancang Bangun Sistem Diagnosis Kerusakan Pada Mobil Menggunakan Metode *Forward Chaining*	Jurnal Universitas Udayana, Bandung 2013	Aplikasi sistem pakar ini dipergunakan mendiagnosis kerusakan mobil diperoleh dari input, berupa gejala awal pada mobil .Sistem ini memberikan saran atas kerusakan yang telah didiagnosa. mempermudah mekanik dan user dalam mendignosa
Arga Dian Setyo Wicaksono	Sistem Pakar Analisa Penyakit Ikan Lele Berbasis Web Menggunakan Metode *Forward Chaining*	Jurnal STEKOM, Semarang 2013	Kurangnya jumlah para ahli atau pakar yang ada di lingkungan sekitar semakin memicu tingkat kegagalan penyakit dan virus. Sistem ini bertujuan mempermudah dalam mengiagnosa penyakit ikan lele

2.4 Metode Pengembangan Sistem

Metode yang digunakan dalam pengembangan sistem ini menerapkan metode *prototyping*. Metode *prototyping* sebagai suatu paradigma dimulai dengan pengumpulan kebutuhan, pengembang bertemu dengan pengguna dan mengidentifikasikan objektif keseluruhan dari perangkat lunak, selanjutnya mengidentifikasi segala kebutuhan yang diketahui secara garis besar dimana definisi-definisi lebih jauh merupakan keharusan, kemudian dilakukan perancangan kilat, lalu diakhiri dengan evaluasi. (Pressman 2012:50). Gambar 2.5 berikut adalah gambar model *prototyping*:



Gambar 2.4 Sistem Model *Prototyping*

Keterangan:

a) Analisis dan Definisi Persyaratan

Proses pengumpulan kebutuhan dilakukan secara intensif untuk mespesifikasikan kebutuhan perangkat lunak agar dapat dipahami perangkat lunak seperti apa yang dibutuhkan oleh *user*. Spesifikasi kebutuhan perangkat lunak pada tahap ini perlu untuk didokumentasikan.

b) Perancangan Sistem

Perancangan sistem dan perangkat lunak adalah proses multilangkah yang fokus pada desain pembuatan program perangkat lunak termasuk struktur data, arsitektur perangkat lunak, representasi antarmuka, dan prosedur pengkodean. Tahap ini mentranslasi kebutuhan perangkat lunak dari tahap analisis kebutuhan ke representasi desain agar dapat diimplementasikan menjadi program pada tahap selanjutnya.

c) Implementasi dan Pengujian Unit

Perancangan sistem dan perangkat lunak ditranslasikan ke dalam program perangkat lunak. Hasil dari tahap ini adalah program komputer sesuai dengan desain yang telah dibuat pada tahap desain.

d) Integrasi dan Pengujian Sistem

Pengujian fokus pada perangkat lunak secara dari segi lojik dan fungsional dan memastikan bahwa semua bagian sudah diuji. Hal ini dilakukan untuk meminimalisir kesalahan (*error*) dan memastikan keluaran yang dihasilkan sesuai dengan yang diinginkan.

e) Operasi dan Pemeliharaan

Tidak menutup kemungkinan sebuah perangkat lunak mengalami perubahan ketika sudah dikirimkan ke *user*. Perubahan bisa terjadi karena adanya kesalahan yang muncul dan tidak terdeteksi saat pengujian atau perangkat lunak harus beradaptasi dengan lingkungan baru. Tahap pendukung atau pemeliharaan dapat mengulangi proses pengembangan mulai dari analisis spesifikasi untuk perubahan perangkat lunak yang sudah ada, tapi tidak untuk membuat perangkat lunak baru.

2.5 UML *(Unified Modeling Language)*

2.5.1 Pengertian UML

Widodo dan Herlawati (2011:15) menjelaskan bahwa dalam proses iteratif pengembangan melakukan beberapa langkah berulang-ulang dan setiap waktu berfokus pada bagian-bagian yang berbeda dari sistem. Alat-alat yang digunakan dalam suatu metodologi umumnya berupa suatu gambar diagram atau gambar grafik.

2.5.2 Bagian-bagian UML

Bagian-bagian utama dari UML adalah *view*, diagram, model elemen, dan *general mechanism*. Diagram berbentuk grafik yang menunjukkan 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:

1) *Use case Diagram*

   Use case adalah abstraksi dari interaksi antara sistem dan actor. *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 client.

2) *Class Diagram*

   Class adalah dekripsi kelompok obyek-obyek dengan *property*, perilaku (operasi) dan relasi yang sama. Sehingga dengan adanya *class diagram* dapat memberikan pandangan global atas sebuah sistem. Hal tersebut tercermin dari *class-class* yang ada dan relasinya satu dengan yang lainnya.

3) *Activity Diagram*

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

4) *Sequance Diagram*

   Sequence diagram (diagram urutan) adalah suatu diagram yang memperlihatkan atau menampilkan interaksi-interaksi antar objek berupa pesan/*message*.

2.5.3 Tujuan dan Keunggulan UML

Tujuan UML adalah memodelkan suatu sistem (bukan hanya perangkat lunak) yang menggunakan konsep berorientasi *object*, menciptakan suatu bahasa pemodelan yang dapat digunakan baik oleh manusia maupun mesin. Keunggulan menggunakan UML dibandingkan menggunakan metodologi terstruktur:

1) *Uniformity*

   Pengembang cukup menggunakan satu metodologi dari tahap analsis hingga perancangan. Memungkinkan merancang komponen antarmuka secara terintegrasi bersama perancangan perangkat lunak dan perancangan struktur data.

2) *Understandability*

   Kode yang dihasilkan dapat diorganisasi kedalam kelas-kelas yang berhubungan dengan masalah sesungguhnya sehingga lebih mudah untuk dipahami.

3) *Stability*

   Kode program yang dihasilkan relatif stabil sepanjang waktu, karena mendekati permasalahan yang sesungguhnya.

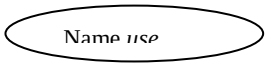
4) *Reusability*

   Dengan metodologi berorientasi objek, dimungkinkan penggunaan ulang kode, sehingga pada akhirnya akan sangat mempercepat waktu pengembangan perangkat lunak (atau sistem informasi).

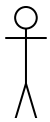
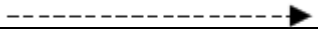
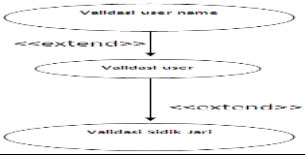
2.5.4 Simbol-simbol pada UML

Simbol-simbol yang terdapat dalam diagram UML dapat dilihat pada tabel 2.3 sebagai berikut :

Tabel 2.3 Simbol-Simbol Pada Diagram UML

Simbol	Deskripsi
Use case 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, biasanya akan diterangkan dengan menggunakan kata kerja di awal di awal frase nama *use case*

Tabel 2.3 (Lanjutan)

Aktor  Nama Aktor	Orang, proses, atau sistem lain yang berinteraksi dangan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendidri, jadi walaupun simbol dari aktor adalah orang, tapi aktor belum tentu merupakan orang, biasanya akan dinyatakan menggunakan kata benda di awal frase nama aktor.
Asosiasi 	Komunikasi antara actor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.
Extensi <<extend>> 	Case tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan, misal 
Uses 	Digunakan sebagai kegiatan utama atau syarat menuju *use case* berikutnya.

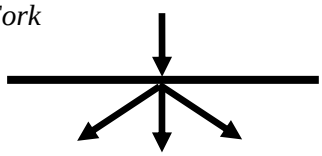
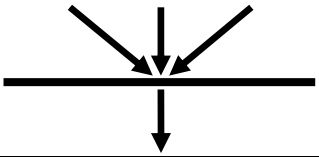
2.5.5 Activity Diagram

Activity Diagram menggambarkan aliran kerja atau aktivitas dari sebuah sistem atau proses bisnis bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem. (Rosa dan Salahudin 2011:134). Simbol-simbol yang digunakan untuk pembuatan *activity diagram* dapat dilihat pada tabel 2.4 sebagai berikut

Tabel 2.4 Simbol *Activity Diagram*

Simbol	Deskripsi
Status awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki status awal.
Aktivitas 	Aktivitas yang dilakukan sistem. Aktivutas biasanya diawali dengan kata kerja.
Percabangan / *decision* 	Asosiasi penggabungan dimana lebih dari satu aktivitas.

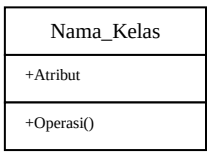
Tabel 2.4 (Lanjutan)

Fork 	Digunakan untuk menunjukan kegiatan yang dilakukan secara paralel.
Penggabungan / *Join* 	Ditunjukan untuk menunjukan kegiatan yang digabungkan.
End Point 	Mengakhiri aktivitas sistem.

2.5.6 *Class Diagram*

Class diagram adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. Class menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut *(metoda/fungsi)*. *Class diagram* menggambarkan struktur dan deskripsi class, *package* dan beserta hubungan satu sama lain seperti containment, pewarisan, asosiasi, dan lain-lain. Simbol-simbol yang digunakan untuk pembuatan *class diagram* dapat dilihat pada tabel 2.5 sebagai berikut:

Tabel 2.5 Bagan *Class Diagram*

Simbol	Keterangan
Kelas 	Kelas pada struktur
Interface  Nama *Interface*	Metode pada *interface* yang digunakan pada suatu kelas sama persis dengan yang ada pada *interface.*

Tabel 2.5 (Lanjutan)

Asosiasi	Relasi antara kelas dengan makna umum**.**
Asosiasi berarah	Relasi antara kelas dengan makna kelas yang satu digunakan pada kelas lain.

2.5.7 Sequace Diagram

Sequence diagram menjelaskan interaksi objek yang disusun dalam suatu urutan waktu. Diagram ini secara khusus berasosiasi dengan *use case*. *Sequence diagram* memperlihatkan tahap demi tahap apa yang sebenarnya terjadi untuk menghasilkan sesuatu didalam *use case*. Penjelasan *sequence diagram* dapat dilihat pada tabel 2.6:

Tabel 2.6 Bagan *Sequance Diagram*

Simbol	Keterangan
Actor	Prilaku didalam *sequence Diagram*
Object	Menambahkan Objek baru pada diagram
Object messege	Menggambarkan pesan (*message*) antar dua objek
Messege to self	Menggambarkan pesan (*message*) yang menuju dirinya sendiri
RETURN MESSEGE	Menggambarkan pengembalian dari pemanggilan prosedur

2.6 Basis Data (*Database*)

Kumpulan terpadu dari elemen data logis yang saling berhubungan. Basis data data mengkonsolidasi banyak catatan yang sebelumnya disimpan dalam file terpisa atau merupakan suatu kumpulan data yang berhununga secara logis dan

deskripsi data terebut yang dirancang untuk memenuhi informasi yang dibutuhkan oleh suatu organisasi. Artinya, basis data merupakan tempat penyimpanan data besar yang dapat digunakan oleh banyak pengguna. Seluruh item basis data tidak lagi dimiliki oleh satu departemen, tetapi menjadi sumber daya perusahaan yang dapat digunakan bersama. (Indrajani, 2009:2).

Tori-teori yang perlu diperhatikan untuk merancang *database* adalah:

1) *Entitas*

   Entitas adalah suatu objek yang dapat dibedakan dengan yang lainnya
2) *Atribut*

   Atribut adalah karakteristik yang menjadi ciri *entitas*
3) *Field*

   Field adalah suatu informasi mengenai suatu *entitas* yang mempunyai arti
4) *Record*

   Record adalah kumpulan dari suatu *field* informasi mengenai *entitas* tertentu atau kumpulan dari item data yang saling berhubungan
5) *File*

   File adalah kumpulan *record* yang saling berhubungan

2.7 Konsep Normalisasi

Setiap *field* atau tabel selalu terdapat kunci (*key*) dari tabel tersebut berupa suatu atribut (*field*) atau beberapa yang dapat mewakili *record*, kunci tersebut dibagi menjadi beberapa jenis yaitu :

a) Kunci Kandidat (*Candidate Key*)

   Candidate Key adalah suatu atribut atau satu set minimal atribut yang mengidentifikasi secara unik suatu kejadian spesifik dan dapat mewakili setiap kejadian dari suatu *record* jika suatu *candidate key* berisi lebih satu atribut, maka disebut kunci gabungan (*Composite Key* (Kristanto 2002:19).

b) Kunci Primer (*Primary Key*)

Primary Key adalah satu atribut atau satu set minimal atribut yang mengidentifikasi secara unik suatu kejadian spesifik dan dapat diwakili setiap kejadian dari satu *entity*. (Kristanto 2002:19).

c) Kunci Alternatif (*Alternative Key*)

Alternatif Key adalah *candidate key* yang tidak dipakai sebagai *primary key*. Kerap kali kunci alternative dipakai sebagai kunci pengurutan dalam laporan misallnya (Kristanto 2002:21).

d) Kunci Tamu (*Foreign Key*)

Foreign Key dalah satu atribut atau satu set atribut yang melengkapi satu *relationship* (hubungan) yang menunjukan ke atribut induknya. (Kristanto 2002:21).

e) Kunci Super (*Super Key*)

Super Key adalah satu atau lebih atribut yang dapat membedakan semua baris data *(record)* dalam tabel secara unik. (Ema Utami dan Sukirno 2005:73).

2.8 Jenis-jenis Hubungan

Relasi antar dua file atau tabel dikategorikan menjadi 4 macam:

a) Satu ke satu (*one to one*)

Setiap entitas pada himpunan entitas A berhubungan dengan paling banyak satu entitas pada himpunan entitas B, dan begitu sebaliknya setiap entitas pada himpunan entitas B berhubungan dengan paling banyak satu entitas pada himpunan entitas A. (Sukirno 2005:73).

b) Satu ke banyak (*one to many*)

Setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B, dan tidak sebaliknya dimana setiap entitas pada himpunan entitas B berhubungan dengan paling banyak satu entitas pada himpunan entitas A. (Sukirno 2005:73).

c) Banyak ke satu (*many to one*)

Setiap entitas pada himpunan entitas A berhubungan dengan paling

banyak satu entitas pada himpunan entitas B, dan tidak sebaliknya dimana setiap entitas pada himpunan entitas B dapat berhubungan dengan banyak entitas pada himpunan entitas A. (Sukirno 2005:73).

d) Banyak ke banyak (*many to many*)

Setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B, dan sebaliknya dimana setiap entitas pada himpunan entitas B dapat berhubungan dengan banyak entitas pada himpunan entitas A. (Sukirno 20 05:73).

2.9 Software Yang Digunakan

Komputer membutuhkan software untuk beroperasi dan membutuhkan sistem operasi atau program-program untuk membuat komponen-komponen komputer bekerja secara baik. Merupakan perangkat yang dapat dilihat oleh mata, tetapi tidak dapat diraba. *Software* juga sering digunakan untuk menunjukkan semua program yang dapat dipakai dalam sistem komputer. Dalam pengertian yang sempit, istilah ini menunjuk pada sebuah program yang dapat mempermudah pemakai dari berbagai jenis komputer untuk mendayagunakan *hardware* dengan baik. (Kristanto 2003:89).

Software dapat digolongkan menjadi beberapa kelompok, yaitu:

1) Sistem Operasi, seperti misalnya program Microsoft Windows, LINUX, Novel Netware, Symbian dan lain sebagainya.
2) Aplikasi, seperti misalnya Microsoft Office, Corel Draw, PhotoShop dan lain sebagainya.
3) *Utility*, seperti misalnya Norton Utilities, Disk Doctor, dan lain-lain.
4) Bahasa Pemrograman, seperti misalnya Java, Visual FoxPro, Visual Basic, Borland Delphi, Bahasa C++, Pascal dan lain sebagainya.
5) Perangkat *software* yang digunakan dalam pembuatan laporan ini adalah Java NetBeans IDE 6.0, AppServ phpMyAdmin