BAB II
LANDASAN TEORI

2.1 Konsep Dasar Sistem

Beberapa hal yang perlu diperhatikan untuk memahami konsep dasar sistem diantaranya adalah definisi sistem dan elemen dasar yang membentuk sistem tersebut. Sistem harus memiliki tujuan dan sasaran yang jelas.

2.1.1 Definisi Sistem

Terdapat dua kelompok pendekatan dalam mendefinisikan sistem, yaitu yang menekankan pada prosedurnya dan yang menekankan pada komponen atau elemennya (Kristanto, 2003):

1. Pendekatan sistem pada prosedur

   Sistem merupakan jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau menyelesaikan suatu sasaran tertentu.

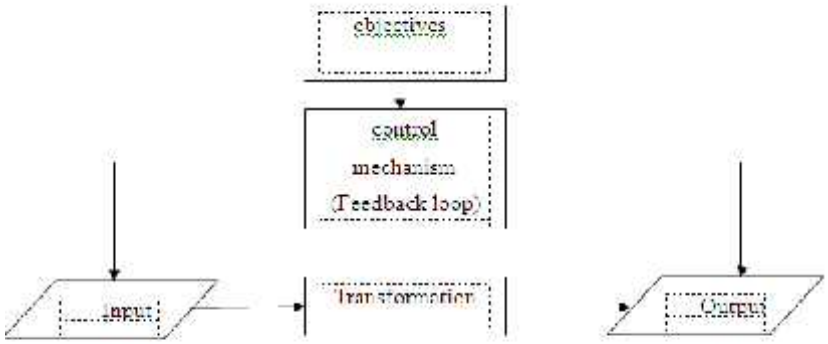
2. Pendekatan sistem yang menekankan pada elemen atau komponen

   Sistem merupakan kumpulan dari elemen-elemen yang berinteraksi untuk mencapai suatu tujuan tertentu. Pendekatan sistem yang merupakan kumpulan dari elemen-elemen atau komponen-komponen atau subsistem-subsistem merupakan definisi yang lebih luas dan lebih banyak diterima karena pada kenyataannya suatu sistem terdiri dari beberapa subsistem atau sistem-sistem bagian. Komponen-komponen atau subsistem-subsistem dalam suatu sistem tidak dapat berdiri sendiri, semuanya saling berinteraksi dan saling berhubungan membentuk satu kesatuan sehingga sasaran sistem dapat tercapai.

Sistem yang baik harus memiliki tujuan dan sasaran yang tepat karena hal ini akan sangat menentukan dalam mendefinisikan masukan (*input*) yang dibutuhkan sistem dan juga keluaran (*output*) yang dihasilkan.

2.1.2 Elemen Dasar Sistem

Hubungan antara elemen-elemen dalam sistem dapat dilihat pada gambar berikut (Jogiyanto, 2001):



Gambar 2.1 Elemen-Elemen Sistem

Keterangan :

1. Objectives : Tujuan yang ingin dicapai.
2. Control Mechanism : Memonitor proses transformasi untuk menjamin bahwa sistem memenuhi tujuannya.
3. Input : Data yang dimasukkan kedalam sistem untuk diproses.
4. Transformation : Proses yang mengolah masukan sistem menjadi keluaran sistem.
5. Output : Keluaran sistem yang menghasilkan laporan dari proses yang telah dilakukan.

2.2 Sistem Pakar (*Expert sistem*)

Perkembangan teknologi komputer dewasa ini semakin pesat baik perangkat keras maupun perangkat lunak, sehingga hampir sebagian pekerjaan manusia kini telah dapat diselesaikan dengan komputer. Dengan demikian, dapat dikatakan bahwa komputer merupakan alat bantu manusia dalam menyelesaikan pekerjaannya. Salah satu alasan mengapa komputer lebih cenderung dikatakan

sebagai alat bantu manusia adalah kecepatan dan ketepatan prosesnya lebih dapat diandalkan. Keinginan manusia untuk menciptakan sesuatu yang baru dimana dapat membantu meringankan beban pekerjaan terus-menerus dilakukan. Hal ini dikarenakan begitu banyaknya kemudahan-kemudahan yang ditawarkan komputer, baik dari segi ketepatan maupun kecepatan informasi.

Kecerdasan Buatan merupakan salah satu bidang dalam ilmu komputer yang ditujukan pada pembuatan *software* dan *hardware* yang dapat berfungsi sebagai sesuatu yang dapat berfikir seperti manusia. Dengan memahami mekanisme penalaran seperti manusia, diharapkan komputer benar-benar merupakan alat bantu yang berguna dalam memecahkan masalah yang memerlukan penalaran.

Salah satu bagian dari kecerdasan buatan yang sedang mengalami perkembangan akhir-akhir ini adalah sistem pakar *(expert sistem),* yaitu suatu sistem yang dirancang untuk dapat menirukan keahlian seorang pakar dalam menjawab pertanyaan dan memecahkan suatu masalah. Sistem pakar akan memberikan pemecahan suatu masalah yang didapat dari dialog dengan pemakai. Dengan bantuan Sistem Pakar seseorang yang bukan pakar/ahli dapat menjawab pertanyaan, menyelesaikan masalah serta mengambil keputusan yang biasannya dilakukan oleh seorang pakar.

Ciri-ciri dari Sistem Pakar adalah sebagai berikut :

a. Terbatas pada domain keahlian tertentu.
b. Dapat memberikan penalaran untuk data-data yang tidak pasti.
c. Dapat mengemukakan rangkaian alasan-alasan yang diberikannya dengan cara yang dapat dipahami.
d. Berdasarkan pada kaidah/*rule* tertentu.
e. Dirancang untuk dapat dikembangkan secara terpisah.
f. Pengetahuan dan mekanisme inferensi jelas terpisah.
g. Keluarannya bersifat anjuran.
h. Sistem dapat mengaktifkan kaidah secara searah yang sesuai, dituntun oleh dialog dengan pemakai.

2.2.1 Konsep Dasar Sistem Pakar

Turban (2005) menyatakan bahwa konsep dasar dari suatu sistem pakar mengandung beberapa unsur/elemen, yaitu keahlian, ahli, pengalihan keahlian, inferensi, aturan dan kemampuan menjelaskan.

Keahlian adalah pengetahuan yang spesifik yang didapatkan dari training, membaca dan pengalaman. Ahli/pakar adalah derajat atau *level* dari keahlian. Pengalihan keahlian yang dimaksud adalah pengaliahan keahlian dari pada ahli ke komputer untuk kemudian dialihkan lagi ke orang lain yang membutuhkan baik orang awam maupun untuk para pakar sebagai asistennya.

Fitur khas dari sistem pakar adalah kemampuan untuk *reasoning*. Keahlian disimpan dalam *knowledge base* dan sistem memiliki akses dalam *database*, maka komputer diprogram untuk dapat berinferensi. Inferensi ini dilakukan oleh komponen yang disebut dengan *inference engine* (mesin inferensi) yang didalamnya terdapat prosedur-prosedur yang berkaitan dengan penyelesaian masalah.

Fitur unik lain dari sistem pakar adalah kemampuan untuk menjelaskan nasehat atau rekomendasi yang diberikan. Penjelasan ini dilakukan oleh subsistem yang disebut dengan *justifier* atau *explanation subsistem*.

2.2.2 Kelebihan dan Kekurangan Sistem Pakar

1. Beberapa kelebihan penerapan Sistem Pakar adalah sebagai berikut :

a. Membuat seorang yang awam bekerja seperti layaknya seorang pakar.
b. Meningkatkan produktivitas akibat meningkatnya kualitas hasil pekerjaan, meningkatnya kualitas pekerjaan ini disebabkan meningkatnya efisiensi kerja.
c. Menghemat waktu kerja.
d. Menyerdehanakan pekerjaan.
e. Merupakan arsip terpercaya dari sebuah keahlian, sehingga bagi pemakai Sistem Pakar seolah-olah berkonsultasi langsung dengan pakar, meskipun mungkin pakar telah meninggal.
f. Memperluas jangkauan, dari keahlian seorang pakar. Dimana sebuah Sistem Pakar yang telah disahkan, akan sama saja artinya dengan

seorang pakar yang tersedia dalam jumlah besar (dapat diperbanyak dengan kemampuan yang persis sama), dapat diperoleh dan dipakai dimana saja.

2. Beberapa Kekurangan Sistem Pakar adalah sebagai berikut :

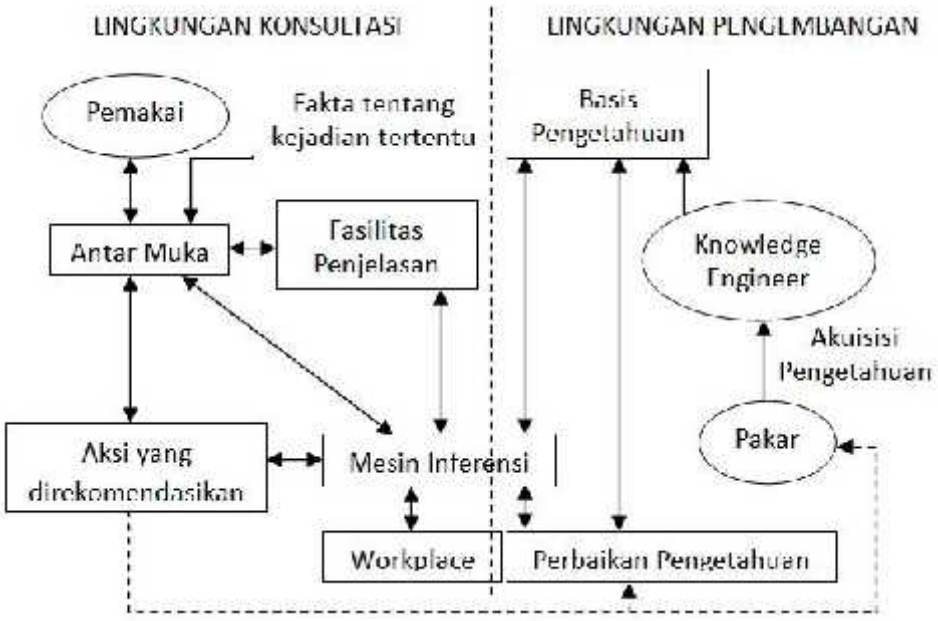
a. Biaya yang sangat mahal membuat dan memeliharanya.

b. Sulit dikembangkan karena keterbatasan keahlian dan ketersediaan pakar.

c. Sistem pakar tidak 100% bernilai benar.

2.2.3 Struktur sistem pakar

Sistem pakar dibagi menjadi 2 bagian utama yaitu lingkungan pengembangan (*defelopment environment*) dan lingkungan konsultasi (*consultation (runtime) environment*). Lingkungan pengembangan digunakan oleh pembangun sistem pakar untuk membangun komponen dan untuk membawa pengetahuan kedalam *knowledge Base*. Lingkungan konsultasi digunakan oleh orang yang bukan ahli untuk mendapatkan pengetahuan dan saran setara pakar.



Gambar 2.2 Diagram Struktur Sistem Pakar (Kusumadewi, 2003)

2.2.4 Komponen Sistem Pakar

Sebuah program Sistem Pakar terdiri atas komponen-komponen sebagai berikut:

1. *Knowledge Acquisition Subsistem*

   Pengetahuan dapat diperoleh dari seorang pakar, teks buku atau laporan penelitian, dengan dukungan dari seorang *knowledge engineer*.

2. Basis Pengetahuan (*Knowledge Base*)

   Basis Pengetahuan merupakan inti program Sistem Pakar dimana basis pengetahuan ini merupakan representasi pengetahuan (*Knowledge Representation*) dari seorang pakar.

3. Mesin Inferensi (*Inferensi Engine*)

   Mesin Inferensi adalah bagian yang mengandung mekanisme fungsi berpikir dan pola-pola penalaran sistem yang akan menganalisis suatu masalah tertentu dan selanjutnya akan mencari jawaban atau kesimpulan yang terbaik. Secara deduktif mesin inferensi memilih pengetahuan yang relevan dalam rangka mencapai kesimpulan. Dengan demikian sistem ini dapat menjawab pertanyaaan pemakai meskipun jawaban tersebut tidak tersimpan secara eksplisit di dalam basis pengetahuan. Mesin Inferensi memulai pelacakannya dengan mencocokan kaidah-kaidah dalam basis pengetahuan dengan fakta-fakta yang ada dalam basis data.

4. *Blackboard*

   Blackboard adalah tempat menyimpan sementara untuk memproses rencana, agenda, solusi dan deskripsi masalah yang didapat dari *Knowledge Base* selama sesi konsultasi.

5. *User*

   User yang dimaksud dalam sistem pakar ini adalah : (1) klien (bukan pakar) yang menginginkan nasehat. (2) *Leaner* (pelajar) untuk mempelajari bagaimana sistem pakar menyelesaikan permasalahan. (3) *Expert sistem Builder* yang meningkatkan *Knowledge Base*-nya. (4) pakar, disini sistem pakar berperan sebagai asistennya.

6. Antarmuka Pemakai *(User Interface)*

   Fasilitas ini digunakan sebagai perantara komunikasi antara pemakai dengan sistem. *User Interface* adalah bagian penghubung antara program sistem pakar dengan pemakai. Pada bagian ini akan terjadi dialog antara program dan pemakai. Program akan mengajukan pertanyaan-pertanyaan berbentuk "ya atau tidak" (*yes or no question*) berbentuk menu pilihan. Program sistem pakar akan mengambil kesimpulan berdasarkan jawaban-jawaban dari pemakai tadi.

7. *Explanation Subsistem*

   Kemampuan penelusuran kebenaran dari konklusi yang didapatkan dari sumber-sumbernya.

8. *Knowledge Refining Sistem*

   Dengan komponen ini pakar mampu untuk menganalisis kinerja dari sistem pakar.

2.3 Variable-Centered Intelligent Rule Sistem (VCIRS)

Dalam sistem pakar salah satu metode yang dapat digunakan adalah *Variable-Centered Intelligent Rule Sistem.* Metode ini merupakan *thesis* master Irfan Subakti yang dipublikasikan pada tahun 2005. Metode ini merupakan sistem berbasis *rule* yang cerdas yang menitikberatkan pada *variabel*, sehingga disebut dengan *Variable-Centered Intelligent Rule Sistem*. Inferensi VCIRS dipertajam oleh analisis *variabel* dan nilai dari urutan dari derajat kepentingan dan tingkat kegunaan dari kasus yang terdapat dalam basis pengetahuan. Dalam VCIRS terdapat istilah *rule, node* dan *variabel*.

Variable-Centered Intelligent Rule Sistem (VCIRS) merupakan teknik persilangan dari *Rule Base Sistem* (RBS) dan *Ripple Down Rule* (RDR).

1. *Rule Base Sistem* (RBS)

   Sistem Berbasis Aturan (SBA- *Rule Base Sistem* (RBS)) adalah sistem yang baik untuk memberikan jawaban dari pertanyaan mengenai *What* (apa), *How* (bagaimana) dan *Why* (mengapa) dari *Rule Base* (RB) selama proses inferensi. Jawaban dan penjelasan disediakan dengan baik. Masalah dengan SBA adalah tidak dapat secara mudah menjalankan proses

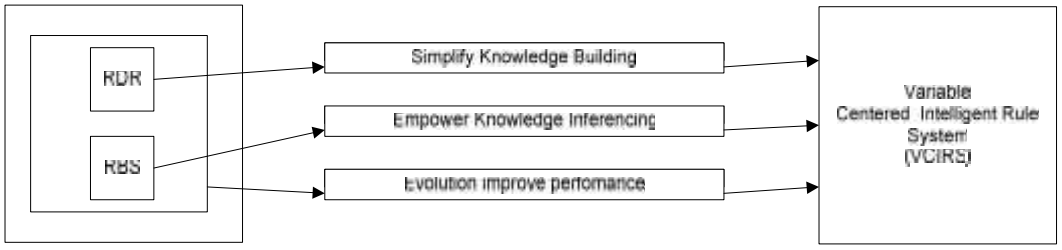
Knowledge Acquisition dan tidak dapat mengupdate *rule* (aturan) secara otomatis. Hanya pakar yang dapat meng*update knowledge Base* (KB) secara manual dengan dukungan dari *Knowledge Engineer.*

2. *Ripple Down Rule* (RDR)

   RDR diciptakan untuk mengatasi permasalahan utama dari sistem pakar. RDR dapat melakukan akuisisi dengan cepat dan sederhana secara ekstrim tampa bantuan dari *Knowledge Engineer*. Pengguna tidak perlu menguji *Rule Base* untuk mendefinisikan *rule* baru. Pakar hanya perlu mendifinisikan *rule* baru secara benar mengklasifikasikan contoh yang diberikan dan sistem dapat menentukan dimana suatu *rule* harus ditempatkan dalam hirarki *rule*nya. Keterbatasan dari RDR adalah kekurangan dalam hal inferensi yang berdayaguna. Tidak seperti SBA yang dilengkapi dengan inferensi melalui *forward chaining* dan *backward chaining*, RDR menggunakan *Depth Firs Seacrh* (DFS) yang memiliki kekurangan dalam hal fleksibelitas dalam hal penjawaban pertanyaan dan penjelasan yang tumbuh dari inferensi yang berdayaguna.

2.3.1 Arsitektur Sistem

Variable-Centered Intelligent Rule Sistem **(VCIRS)** merupakan perkawinan dari *Rule Base Sistem* (RBS) dan *Ripple Down Rule* (RDR). Arsitektur sistem diadaptasi dari RBS dan mengambil keuntungan yang ada dari RDR. Keuntungan tersebut dapat dilihat pada gambar dibawah ini.

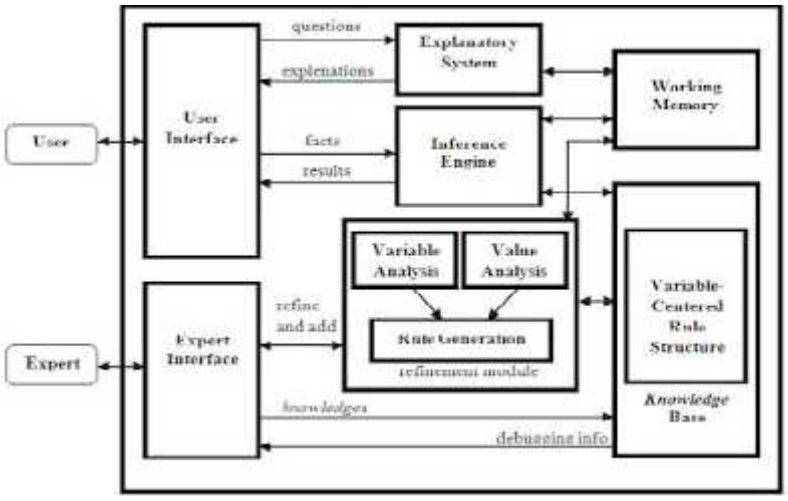


Gambar 2.3 Diagram Metode VCIRS

Istilah "*Intelligent*" dalam VCIRS menekankan pada keadaan sistem ini yang dapat "belajar" untuk meningkatkan kinerja sistem dari pengguna sistem selama pembangunan pengetahuan (melalui analisis nilai) dan penghalusan pengetahuan (dengan pembangkitan *rule*).

Pembangunan pengetahuan disederhanakan, jadi pengguna tidak perlu mempertimbangkan mengenai struktur Basis Pengetahuan dan dapat meng*update* Basis Pengetahuan secara langsung. VCIRS juga membolehkan pengguna untuk memperbaiki atau menambahkan *node/rule* kedalam Basis Pengetahuan yang telah ada. Seperti dalam RDR, perbaikan *rule* adalah pembuatan sebuah *rule* pengecualian untuk membenarkan pengklasifikasian yang salah, *rule* ini diletakkan pada level puncak tree Basis Pengetahuan. Sistem memandu pengguna selama proses pembangunan pengetahuan.

Inferensi pengetahuan dipertajam oleh pengetahuan (yaitu, hasil dari analisis variabel dan nilai) dari urutan derajat kepentingan (*important degree*) dan tingkat penggunaan (*usage rate*) dari kasus pada Basis Pengetahuan. Mekanisme inferensi RBS dibawa kembali dalam VCIRS, sehingga pengguna mendapatkan lebih banyak jawaban dan penjelasan dari inferensi. Hal ini dapat dilihat pada gambar di bawah ini yang menggambarkan arsitektur RBS, gambar dibawah menjelaskan modifikasi dari arsitektur RBS yang menambahkan analisis variabel dan nilai untuk membantu proses pembangkitan *rule* pada modul perbaikan *Knowledge Base* sehingga menjadi arsitektur VCIRS.



Gambar 2.4 Arsitektur VCIRS

Gambar diatas menjelaskan tentang arsitektur VCIRS dimana dipresentasikan oleh pengguna dan menuju ke memori kerja selama pembangunan pengetahuan, disimpan secara permanen ke dalam *Variable-Centered Rule Structure* disaat sistem menyimpan informasi *rule* dan menghitung kejadian dari setiap . Informasi *rule* yang tersimpan tadi digunakan oleh *Variable Analysis* untuk mendapatkan *important degree*. Kejadian dari setiap digunakan oleh *Value Analysis* untuk mendapatkan *usage degree*. *Usage degree* membantu pengguna sebagai garis pedoman selama pembangunan dan inferensi pengetahuan untuk penentuan variabel mana yang diinginkan untuk dikunjungi pertama kalinya. *Usage degree* bersama *important degree* akan mendukung *Rule Generation* untuk memproduksi *rule/node* baru.

Arsitektur sistem dari VCIRS mendukung tiga operasi yang berhubungan dengan KB, yaitu:

1. Pembangunan pengetahuan yang membolehkan pengguna untuk membuat Basis Pengetahuan dari tidak ada sama sekali (*scratch*) atau untuk mengupdate Basis Pengetahuan yang telah ada. VCIRS membangun Basis Pengetahuan baru berdasarkan kasus yang disediakan oleh pengguna.
2. Perbaikan pengetahuan. Ini membolehkan pengguna untuk mendapatkan *important degree* (derajat kepentingan) dan *usage degree* (derajat kegunaan) dari suatu *variabel/node/rule*, atau untuk membangkitkan *rule* baru.
3. Inferensi pengetahuan untuk melakukan inferensi*/reasoning* dari Basis Pengetahuan. Inferensi dapat dilakukan menggunakan metode inferensi RDR atau RBS.

Ada dua pendekatan inferensi yang dipakai dalam operasi ini, yaitu pendekatan RBS dan RDR. Inferensi pengetahuan secara mudah adalah proses pembangunan pengetahuan tanpa aksi yang dilakukan oleh pengguna. Pengguna memasukkan dan sistem berjalan melalui penelusuran proses, saat VCIRS melakukan proses *forward chaining* sederhana. Setiap variabel disimpan bersama dengan posisinya, proses inferensi dapat berjalan sangat cepat karena VCIRS menemukan sebuah variabel, *node* atau *rule* dengan mudah melalui posisi mereka.

Selama proses inferensi, VCIRS memperlakukan sebuah *rule* sebagai rangkaian dari *node* (*rule* dalam RBS). Sehingga mengabaikan isi konklusi dari setiap *node*, kecuali konklusi pada *node* terakhir sebagai konklusi dari rule. Inferensi memperlakukan sebuah *rule* sebagai *rule* (besar) dimana *clause part*-nya mengandung semua clause part dalam setiap *node* dari rangkaian dan *conclusion part*-nya adalah konklusi dari *node* terakhir. Sehingga dari sini, *operator clause* adalah operator AND (dari semua *clauses*), yang juga merupakan jenis operator konklusi jika ada lebih dari satu nilai konklusi dalam suatu *node* (maka ia menjadi *node* terakhir dalam suatu *rule*).

2.3.1.1 Variable-Centered Rule Structure

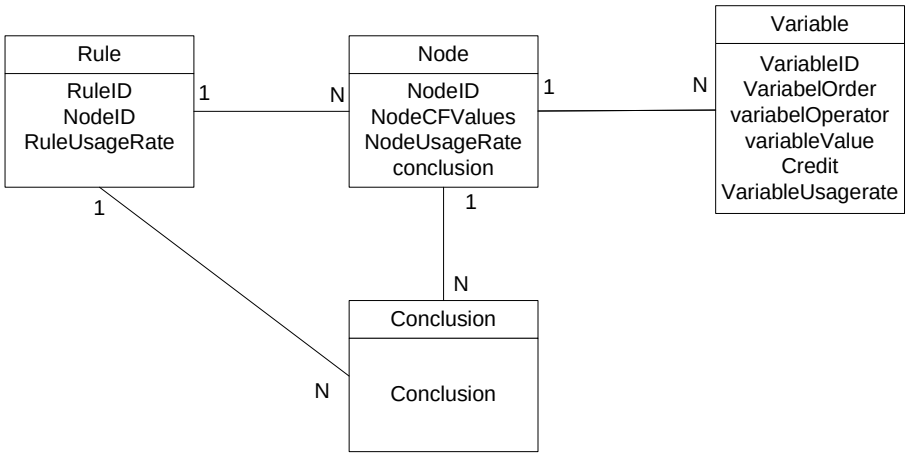
Variable-Centered Rule Structure digunakan untuk merepresentasikan Basis Pengetahuan dan mendukung *Refinement Module* untuk mengelola Basis Pengetahuan yang *up-to-date*. VCIRS juga mencatat kasus-kasus dan kejadiannya. Elemen fundamental dari *Variable-Centered Rule Structure* adalah variabel, yang ditempatkan/dipasang oleh pengguna. VCIRS mengelola secara cermat variabel ini mengenai nilainya, struktur dan kejadiannya. Rangkaian dari variabel membentuk *node,* sedangkan rangkaian dari node menyusun *rule*. Maka *Variable-Centered Rule Structure* mengandung struktur rule dan struktur node yang berpusat pada variabel-variabel.

Gambar 2.5 Node Structure

Gambar 2.5 diatas menjelaskan tentang struktur *node* yang berfungsi menyimpan kasus yang dipresentasikan oleh pengguna dan menghitung kejadian dari setiap kasus, struktur ini serupa dengan sebuah *rule* dalam RB dari RBS. *Struktur node* juga menyimpan kejadian dari variabel-variabel dan *node-node* untuk *usage assignment*. Kasus baru yang disediakan oleh pengguna akan

dimasukkan ke dalam struktur *node* oleh memori kerja dan lalu menggunakan struktur *rule* sebagai bagian dari Basis Pengetahuan yang *up-to-date*.

Setiap kasus terdiri dari kumpulan *field* data seperti digambarkan pada gambar 2.5. Setiap kali pengguna memasukkan kasus-nya, struktur *node* mengelola nilai dan posisinya. Informasi ini akan digunakan dalam usage assignment. Berdasarkan isi dari suatu kasus, VCIRS akan mencari dalam Basis Pengetahuan untuk *node* yang layak. *Node* disini serupa dengan *rule* dalam RBS, yang mengandung satu atau lebih variabel bersamaan dengan nilainya (*clause part*), dan satu atau lebih konklusi (*conclusion part*).



Gambar 2.6 Rule Structure

Gambar diatas menggambarkan struktur rule menyimpan rangkaian dari *node* yang direpresentasikan oleh struktur *node*. Setiap *node* dalam Basis Pengetahuan memiliki rangkaian, yaitu paling tidak ada satu *node* untuk satu *rule*. Kasus yang dimasukkan oleh penguna pertama kali disimpan dalam *Node Structure,* lalu ia akan digunakan dalam *Rule Structure*. ID dari sebuah *rule* sama dengan <*Node ID*> terkecil dari setiap *rule*. Jika *rule* berikutnya memiliki kandidat <*Rule ID*> yang sama maka sistem akan menamainya seperti penamaan pada *node*, nama dari *rule* yang telah ada diikuti dengan nomor serial.

2.3.1.2 Refinement Module

Ada 3 tugas dalam *Refinement Module*, tugas tugas tersebut berguna dalam proses *up-to-date Knowledge Base*. Tugas-tugas tersebut adalah:

1. Analisis variabel. Analisis variabel menentukan manakah variabel/*node* yang paling penting.
2. Analisis nilai. Analisis nilai menetukan seberapa sering sebuah *rule/node/variabel* itu digunakan.
3. Pembangkitan *node*. Pembangkitan *node* adalah hasil dari analisis variabel dan analisis nilai.

Di dalam *Variable-Centered Rule Structure* sistem mengetahui node mana yang di-*shared* (sama-sama menggunakan) oleh berbagai *rule* yang memakai suatu *node,* maka *node* itu akan semakin penting. Pertimbangan yang sama terjadi pada variabel yang di-shared di dalam *node*.

Proses analisa nilai, yang disebut dengan *uses assignment* (pemberian nilai kegunaan), adalah untuk menentukan drajat kegunaan (*Usage Degree*) dari *rule/node*/variabel dalam KB. *Usage assingnment* menggunakan informasi yang disimpan dalam *Variable-Centered Rule Structure*. Ada 3 jenis *Usage Degree*, yaitu:

1. *Variable Usage Rate* (VUR) digunakan untuk mengukur tingkat kegunaan dari suatu variabel di dalam *node* yang sedang dan telah digunakan.

   $VUR_i = Credit_i \times Weight_i$ (2.1)

   $Weight_i = NS_i \times CD_i$

   $CD_i = \frac{VO_i}{TV}$

2. *Node Usage Rate* (NUR) untuk mengukur tingkat kegunaan suatu *node* pada pengeksekusian (firing).

   $NUR_J = \frac{\sum_1^N VUR_{IJ}}{N}$ VURij untuk variabel ke I dalam *node* j (2.2)

3. *Rule Usage Rate* (RUR) yang mengukur tingkat kegunaan suatu *rule* pada pengeksekusian (*firing*).

$$RUR_J = \frac{\sum_1^N NUR_{JK}}{N}$$, NURjk untuk *node* ke j dalam *rule* k (2.3)

Keterangan

$Credit_i$: Kejadian dari variabel I dalam *Node Structure*

$Weight_i$: Menghitung bobot (*Weight*) dari variabel ke *nod*e yang memilikinya

NS_i : Jumlah *node* yang berbagi (*sharing*) variabel i

CD_i : Derajat Kedekatan dari variabel i dalam *node* j

VO_i : Urutan dari variabel I dalam suatu *node*

TV : Total variabel yang dimiliki oleh suatu *node*

Makin besar indikator kegunaan, maka makin bergunalah nilai tersebut dan begitu pula sebaliknya.

Pembangkit *node* bekerja berdasarkan hasil dari analisis variabel dan nilai. Informasi mengenai *shared node*/variabel dari analisis variabel berguna untuk memilih kandidat yang baik untuk membuat kombinasi. *Sahared node*/variabel yang paling tinggi nilainya berarti bahwa ialah yang merupakan *node*/variabel terpenting dalam KB yang ada, karena ia digunakan di banyak tempat pada struktur saat ini. *Usage degree* dengan nilai tertinggi yang diperoleh dari analisis nilai berarti bahwa *node*/variabel tersebut memiliki kejadian (*occurrence*) tertinggi di dalam struktur.

2.3.2 Pembangunan Pengetahuan

Pembagunan pengetahuan dalam VCIRS mengizinkan pakar untuk membangun KB dari tidak ada sama sekali (*scratcha*) atau untuk meng-*update* KB yang telah ada.

2.3.2.1 Pohon Inferensi VCIRS

Berdasarkan isi dari suatu kasus, VCIRS akan mencari *node* yang layak. Adapun algoritma untuk mencari *node* yang layak adalah menemukan *node* yang sempurna yaitu *node* mengandung ciri-ciri dan nilai konklusi yang sama. Yang kedua *node* yang mengandung ciri-ciri yang sama dan juga memiliki nilai

konklusi yang sama, paling tidak *node* mempunyai satu ciri-ciri yang sama paling tidak ada satu nilai konklusi yang sama, dan *node* yang memiliki paling tidak ada satu gejala yang sama.

Sedangkan algoritma untuk membangun basis pengetahuan adalah sebagai berikut : Jika *node* yang layak ditemukan, maka sistem akan memberikan tabel perbedaan yang memuat 2 bagian, yaitu bagian yang berisi kondisi *rule* lama (A) dan bagian yang berisi kondisi *rule* baru (B). Kemudian sistem memberikan beberapa opsi berdasarkan table perbedaan tersebut. Adapun opsi tersebut adalah :

a. Pakar menerima *rule* yang telah ada, tak memasalahkan case baru yang tersedia.
b. *Rule* baru yang akan dihasilkan adalah negasi dari kondisi bagian A yang berbeda pada daftar perbedaan.
c. *Rule* baru yang kan dihasilkan adalah kondisi pada bagian B yang berbeda pada daftar perbedaan.
d. *Rule* baru yang akan dihasilkan adalah gabungan negasi bagian A yang berbeda dan bagian B yang berbeda.
e. *Rule* baru yang akan dihasilkan semuanya berasal dari *case* baru yang tersedia tampa melihat daftar perbedaan, dan letaknya langsung dibawah *root* imajiner. Dengan kata lain ini adalah *rule* yang terletak pada level puncak KB.

Jika node yang layak tidak ditemukan maka sistem akan membuat *node* baru pada level puncak KB (parentnya root). Jika suatu kasus diklasifikasikan tidak benar/salah maka hal yang harus dihasilkan adalah menghentikan *rule* tersebut sehingga tidak digunakan dalam proses inferensi kemudian membangun kasus baru yang benar pada level puncak KB. Untuk menghentikan suatu *rule* cukup ditambahkan suatu tanda padanya yang menandakan bahwa *rule* ini tidak boleh di pakai lagi (artinya konklusinya tidak boleh digunakan, namun *clause*nya masih dapat digunakan oleh *rule-rule* dibawahnya) hal tersebut disebut dengan *stopping rule*.

2.3.3 Faktor Kepastian (*Certainty Factor*)

Dalam menghadapi suatu masalah sering ditemukan jawaban yang tidak memiliki kepastian penuh. Ketidakpastian ini bisa berupa probabilitas yang tergantung dari hasil suatu kejadian. Sistem pakar harus mampu bekerja dalam ketidakpastian. Sejumlah teori telah ditemukan untuk menyelesaikan ketidakpastian, termasuk diantaranya probabilitas klasik (*classical probability*), probabilitas Bayes (*bayesian probability*), teori Dempster-Shafer, teori fuzzy Zadeh dan faktor kepastian (*certainty factor*). Dalam penelitian ini yang digunakan adalah faktor kepastian (*certainty factor*).

Certainty Factor (CF) adalah ukurtan/tingkat kepercayaan seseorang terhadap *rule* yang ada. Untuk mengetahui faktor kepastian oleh pengguna tidaklah mudah, karna sulit bagi pengguna untuk memperkirakan besarnya nilai kepastian terhadap elemen *antecedent* sesuai dengan standar yang diberikan oleh pakar.

$$\text{CFkombinasi} \begin{cases} CF_1 + CF_2\,(1-CF_1) \text{ kedua-duanya } >0 \\ \dfrac{CF_1 + CF_2}{1-\min([CF_1][CF_2])} \\ CF_1 + CF_2\,(1-CF_1) \text{ kedua-duanya } <0 \end{cases} \qquad (2.4)$$

2.3.4 Proses Inferensi

Ada 2 tipe teknik inferensi yang ada yaitu :

1. **Pelacakan Ke Belakang** (*Backward Chaining*) yang memulai penalarannya dari sekumpulan hipotesa menuju fakta-fakta yang mendukung hipotesa tersebut.
2. **Pelacakan Ke Depan** (*Forward Chaining*) yang merupakan kebalikan dari pelacakan ke belakang, yaitu memulai dari sekumpulan data menuju kesimpulan.

Selama proses inferensi, VCIRS memperlakukan sebuah *rule* sebagai rangkaian dari *node* (*rule* dalam RBS). VCIRS mengabaikan isi konklusi dari setiap *node*, kecuali pada *node* terakhir sebagai konklusi dari *rule*. Dari titik pandang ini inferrensia memperlakukan sebuah *rule* sebagai *rule* (besar) dimana

clause part-nya adalah konklusi dari *node* terakhir. Sehingga dari sini, operator *clause* adalah operator AND (dari semua *clauses*), yang juga merupakan jenis operator konklusi jika ada lebih dari satu nilai konklusi dalam suatu *node* (dan lalu ia menjadi *node* terakhir dalam suatu *rule*) (Subakti, 2006).

2.3.5 Up-date Knowledge Base

Proses *up-date* KB terjadi pada *Refinemnt Module*. Dari proses tersebut akan di peroleh *node* baru yang di dapatkan dari kasus yang telah ada. Untuk mendapatkan *node* tersebut yang harus dilakukan adalah proses pembangkit *node*. Dimana pada proses pembangkit *node* akan diperoleh kombinasi variabel-variabel sehingga menghasilkan *node* baru. Algoritma sebagai pembangkit *node* adalah sebagai berikut:

Algoritma pembangkit *node* mengkombinasikan variabel-variabel terpenting yang dihasilkan oleh analisa variabel. Variabel-variabel tersebut disusun berdasarkan urutan relatif variabel yang dihitunng oleh "Algoritma penghitungan urutan relatif variabel". Hasil dari pembangkit *node* diberi karakter "G" sebagai pembeda antara *node* lama dan *node* hasil *up-date* sistem. Kemudian tampilan *node* kepada pakar/admin sebagai konfirmasi sebelum *node* tersebut disimpan sebagai *node* tambahan dalam KB.

Tabel 2.1 Struktur Data Penghitungan Urutan Relatif Variabel

Step	***Current Node (NUR)***	***Variabel Order Queue (VUR)***	***Node Used (NUR)***	***Pre Candidate Variabel (VUR)***	***Candidate Variabel***	***Node Stack***
1	…….	…….	…….	…….	…….	…….
2	…….	…….	…….	…….	…….	…….

"*Current Node*" menyimpan *NodeID* yang sedang diproses, dimulai dari *node* dengan NUR terendah "*Variabel Order Queue*" menyimpan urutan dari variabel-variabel dari *node* yang sedang diproses dalam "*Current Node*", yang prose dimulai dari urutan pertama. "*Node Used*" menyimpan setiap *node* yang berbagi *(sharing)* suatu variabel dalam "*Pre Candidate Variabel*" didasarkan pada VUR

dari *node*. Variabel yang mempunyai VUR terendah akan di ambil lebih dulu. "*Candidate Variabel*" menyimpan urutan relatif variabel. "*Node Stack*" memasukan (*push*) *node* ke *stack*, setelah suatu *node* selesai diproses dalam "*Current Node*".

Kandidat variabel yang diperoleh dari penghitunagan urutan relatif variabel akan digunakan untuk meng-*update* KB sehingga menghasilkan *node* baru dari *rule/node* yang telah ada (lama).

2.3.6 Evaluasi Sistem

Seperti halnya RDR, VCIRS hanya memberikan perhatian pada variabel (*clause part*) pada bagian konklusi. Konklusi tidak mempunyai peranan dalam sistem, konklusi hanyalah hasil yang begitu saja diperoleh. Ini berarti kita dapat mengabaikan konklusi yang tanpa resiko apapun.

Ketidakpentingan konklusi selama proses penulusuran *tree rule* memiliki konsekuensi: tak ada mekanisme untuk melakukan *forward chaining* dan *backward chaining* seperti dalam RBS. RDR tidak menyebutkan mengenai inferensi, sebab RDR ditujukan utamanya untuk akuisisi pengetahuan yang cepat dan sederhana, bukan untuk inferensi. Inferensi dalam RDR dilakukan pada waktu yang sama dengan waktu pengguna melakukan pembangunan pengetahuan dengan penyedian *case* dan mengikuti sitem kerja. Inferensi dalam RDR dapat memilih untuk hanya menelusuri *tree rule* selama operasi tanpa membuat *rule* baru, artinya dia hanya ingin melakukan inferensi (yaitu *forward chaining* sederhana). Fakta bahwa untuk menelusuri *tree* (inferensi) dan meng-*up-date* KB pada waktu yang sama mengandung arti bahwa verifikasi dari KB dapat dilakukan pada saat berjalannya sistem. VCIRS mewarisi kedua keuntungan ini, yaitu akuisisi pengetahuan yang mudah dan sederhana dan verifikasi sambil jalan (*verification-on-the-fly*). Sebagai tambahan untuk pendekatan penelusuran *tree* seperti RDR untuk membuat pembangunan (*up-date*) pengetahuan mudah, VCIRS juga menyediakan mekanisme untuk melakukan tranformasi KB sehingga pengguna dapat melakukan inferensi RBS yang berdayaguna. Yang dibutuhkan oleh VCIRS adalah pengguna harus konsisten dalam menggunakan variabel ID

baik pada *clause* maupun pada *conclusion part*, dan VCIRS mampu untuk mengelola konsistensi *rule-rule* logik dalam keseluruhan KB.

Dibandingkan dengan RDR, VCIRS memiliki dayaguna yang sama dalam pembangunan pengetahuan. VCIRS juga menyimpan ruang untuk struktur *rule* yang tidak menyimpan setiap rangkaian *node*nya. VCIRS cukup mengingat posisi dari setiap *node* dan selanjutnya adalah mudah untuk menyusun ulang *rule* sebagai rangkainan dari *node-node*. VCIRS menyimpan setiap kejadian dari *case* dalam *node structure* untuk setiap *node*, membantu sistem menghasilkan *rule* baru dari perbaikan pengetahuan, dimana RDR tidak dapat melakukan hal ini.

Dibandingkan denagn RBS, VCIRS dapat melakukan *verification-on-yhe-fly* pengetahuan, dimana RBS tidak dapat melakukannya, verifikasi harus dilakukan oleh pakar secara manual yang amat memakan waktu dan mempunyai kecenderungan untuk terjadinya inkonsistensi yang lebih buruk. Dalam RBS, *rule* yang inkonsistensi cenderung akan berdampak pada *rule-rule* yang lain. Hal ini bukan merupakan hal yang serius dalam VCIRS, karena *rule* yang inkonsisten hanya berakibat pada *sharing nodes (rules*) yang dimiliki oleh beberapa *rule* saja, bukan keseluruhan *rule*. Walaupun beberapa *nodes* (*rules*) memiliki nilai-nilai yang sama dengan yang lain, kecuali *node-node* dalam rangkaian yang sama, maka tak ada apapun yang terjadi.

VCIRS selalu melakukan kalkulasi ulang untuk VUR pada variabel yang terkait, jika suatu variabel dimasukkan kedalam *Variabel Rule Structure*. VCIRS juga mengkalkulasi ulang NUR dari semua *node* yang menggunakan variabel tersebut, dan mengupdate RUR dari semua *rule* yang menggunakan *node* ini. Hal tersebut akan sangat terlihat membosankan, namun berguna untuk memandu pengguna dalam prose pembangunan dan inferensi pengetahuan. Ini juga berguna untuk mendukung pembangkitan *node*.

Disaat melakukan pembangkitan *node*, VCIRS meminta pengguna untuk mengkonfirmasikan *rule/node* yang sedang dibangkitkan, sebab pembenaran logik (semantik) dari pembangkitan *rule/node* masih memerlukan pertimbangan pengguna. Sistem hanya menghasilkan rangkaian alternatif dari variabel-variabel yang penting untuk membentuk *node* baru dan rangkaian dari *node* untuk membentuk *rule* masing-masing menurut urutan relatif dari variabel dan *node*.

Walaupun VCIRS mengatasi inferensi dari RDR, *Algoritma Rate* yang dikembangkan oleh *forgy* masih memiliki kinerja yang lebih baik dibangdingkan dengan langkah-langkah (*cycles*) yang harus dilakukan sebelum mendapatkan hasil. *Algoritma Rate* digunakan dalam kebanyakan RBS kinerja tinggi, sebab ia adalah metode yang sangat efisien untuk permasalahan pengenalan pola. Kelemahannya adalah ia memiliki kompleksitas ruang tinggi (*hight space complexity)*. VCIRS yang memiliki *Variabel Centered Rule Structure* menyimpan setiap posisi dari setiap variabel dalam suatu *node* dan *rule* menyediakan kinerja yang baik dan masuk akal, tidak terlalu sempurna seperti halnya *Algoritma Rate*, namun masalah kompleksitas ruang cendrung dapat lebih rendah diwujudkan.

2.4 Pemodelan Analisis

Model analisis merupakan serangkain model yang mempresentasikan teknis yang pertama dari sistem. Model analisis harus dapat mencapai tiga sasaran utama, yaitu: (1) untuk menggambarkan apa yang dibutuhkan oleh pelanggan, (2) untuk membangun dasar bagi pembuatan desain perangkat lunak dan (3) untuk membatasi serangkaian persyaratan yang dapat divalidasi begitu perangkat lunak dibangun (Pressman, 2002).

Pada inti model ada kamus data-penyimpanan yang berisi deskripsi dari semua objek data yang dikonsumsi atau diproduksi oleh perangkat lunak. *Entity Relationship Diagram* (ERD) menggambarkan hubungan antar objek data. Atribut dari masing-masing objek data yang ditulis pada ERD dapat digambarkan dengan menggunakan deskripsi objek data. *Data Flow Diagram* (DFD) adalah sebuah teknis grafis yang menggambarkan aliran informasi dan transformasi yang diaplikasikan pada saat data bergerak dari *input* menjadi *output*. DFD mempunyai dua tujuan: (1) untuk memberikan indikasi mengenai bagaimana data ditransformasikan pada saat bergerak melalui sistem dan (2) untuk menggambarkan fungsi-fungsi yang mentransformasi aliran data (Pressman, 2002).

2.5 Kejahatan Dunia Maya (*Cybercrime*)

Dalam dua dokumen Konferensi PBB mengenai *The Prevention of Crime and the reatment of Offenders* di Havana, Cuba pada tahun 1990 dan di Wina, Austria pada tahun 2000, ada dua istilah yang dikenal, yaitu *cybercrime* dan *computer related crime.* Dalam *back ground paper* untuk lokakarya Konferensi PBB X/2000 di Wina, Austria istilah *cybercrime* dibagi dalam dua kategori. Pertama, cybercrime dalam arti sempit disebut *computer crime*. Kedua, *cybercrime* dalam arti luas disebut computer *related crime.*

Dengan demikian *cybercrime* meliputi kejahatan, yaitu yang dilakukan dengan (1). menggunakan sarana-sarana dari sistem atau jaringan komputer (2). Di dalam sistem atau jaringan komputer dan (3). Terhadap sistem atau jaringan komputer. Dari definisi tersebut, maka dalam arti sempit *cybercrime* adalah *computer crime* yang ditujukan terhadap sistem atau jaringan komputer, sedangkan dalam arti luas, *cybercrime* mencakup seluruh bentuk baru kejahatan yang ditujukan pada komputer, jaringan komputer dan penggunanya serta bentuk-bentuk kejahatan tradisional yang sekarang dilakukan dengan menggunakan atau dengan bantuan peralatan komputer (*computer related crime*), (Tim Perundang-Undangan dan Pengkajian Hukum Bank Indonesia, 2006).

Andi Hamzah memberikan batasan dan definisi dari kejahatan komputer tidak jauh berbeda yang dikemukakan oleh NPA: bahwa kejahatan di bidang komputer secara keseluruhan dapat diartikan sebagai penggunaan komputer secara illegal (Hamzah, 1989). Semua perumusan atau batasan yang diberikan mengenai kejahatan komputer (*computer crime*), atau penyalahgunaan komputer (*computer misuse*) tersebut secara umum dapat disimpulkan bahwa perbuatan atau tindakan yang dilakukan dengan menggunakan komputer sebagai alat/sarana untuk melakukan tindak pidana atau komputer itu sendiri sebagai objek tindak pidana.

Dalam membicarakan pengaturan teknologi informasi yang juga penting untuk diperhatikan adalah masalah penegakan hukum (*law enforcement*). Penegakan hukum merupakan isu yang penting dalam konteks pembuatan peraturan perundang-undangan. Hal ini karena penyelesaian masalah hukum terhadap sesuatu yang baru tidak hanya cukup berhenti dengan dibentuknya Undang-Undang.

Menjawab tuntutan dan tantangan komunikasi global lewat internet, undang-undang yang diharapkan adalah perangkteat hukum yang akomodatif terhadap perkembangan serta antisipatif terhadap permasalahan, termasuk dampak negatif penyalahgunaan internet dengan berbagai motivasi yang dapat menimbulkan korban-korban seperti kerugian materi dan non materi. Saat ini, Indonesia memiliki undang-undang yang mengatur tentang Informasi Transaksi Elektronik yaitu, Undang Undang Nomor 11 Tahun 2008.

2.5.1 Jenis Jenis Kejahatan

a. Kejahatan Ringan :

1. Jenis Kejahatan :Penghinaan / Pencemaran nama baik [Pasal 27 ayat 3]

   Ciri-Ciri :

   a. Mempublikasikan artikel berisi penghinaan terhadap suatu perusaahan/ instansi melalui situs web/ blog atau email.
   b. Mempublikasikan artikel berisi penghinaan terhadap orang lain melalui situs web/ blog atau email.
   c. Memanipulasi dan mempublikasikan identitas milik pribadi seseorang yang bersifat fitnah melalui media internet.
   d. Mempublikasikan artikel berisi sindiran maupun pelecehan terhadap suatu perusaahan/ instansi melalui situs web/ blog atau email.

2. Jenis Kejahatan :Pemerasan/ Pengancaman [Pasal 27 ayat 4]

   Ciri-Ciri :

   a. Mengirimkan pesan yang berupa ancaman kepada pihak tertentu.
   b. Memanfaatkan metode DDos (distributed denial of service) untuk menyerang sebuah situs web, dengan cara meminta sejumlah uang kepada pemilik situs untuk dapat menghindari serangan yang akan dilakukan ke situs tersebut.
   c. Mengirimkan pesan yang berupa pemerasan kepada pihak tertentu.

3. Jenis Kejahatan :Perjudian [Pasal 27 ayat 2]

   Ciri-Ciri :

   a. Melakukan perjudian online melalui sebuah situs web.
   b. Membuat perjudian online.

c. Mendistribusikan atau menyalurkan dan menjadi perantara dapat diaksesnya perjudian online melalui sebuah situs web.

4. Jenis Kejahatan : SARA [Pasal 28 ayat 2]

   Ciri-Ciri :

   a. Mempublikasikan berita yang menimbulkan rasa kebencian/ permusuhan terhadap seorang atau kelompok masyarakat tertentu berdasarkan atas suku, agama, ras dan antar golongan (SARA).
   b. Mempublikasikan berita yang menimbulkan rasa kebencian/ permusuhan terhadap seorang atau kelompok masyarakat tertentu terhadap pemerintah atau instansi.

5. Jenis Kejahatan : Penyebaran berita palsu [Pasal 28 ayat 1]

   Ciri-Ciri :

   a. Menyebarkan berita bohong dan menyesatkan yang mengakibatkan kerugian konsumen.
   b. Menyebarkan berita menyesatkan yang mengakibatkan kerugian orang lain konsumen.

b. Kejahatan Sedang

1. Jenis Kejahatan : Penyalahgunaan [Pasal 30 ayat 1]

   Ciri-Ciri :

   Mengakses sistem orang lain tanpa sepengetahuan si pemilik dengan cara apapun.

2. Jenis Kejahatan : Penyalahgunaan [Pasal 30 ayat 2]

   Ciri-Ciri :

   Mengakses sistem orang lain tanpa sepengetahuan si pemilik dengan cara apapun dengan tujuan, mencuri data maupun informasi.

3. Jenis Kejahatan : Penyalahgunaan [Pasal 30 ayat 3]

   Ciri-Ciri :

   a. Mengakses sistem orang lain tanpa sepengetahuan si pemilik dengan cara apapun dengan melanggar, menerobos, melampaui atau menjebol sistem pengamanan.

b. Melakukan pencurian data nasabah bank demi mendapatkan keuntungan pribadi melalui media internet dan bertransaksi secara online.
c. Bekerja sama dengan pihak perbankan untuk melakukan data nasabah bank.

4. Jenis Kejahatan : Hacker [Pasal 32 ayat 1]

   Ciri-Ciri :

   a. Mengubah suatu Informasi Elektronik dan/atau Dokumen Elektronik milik Orang lain atau milik publik.
   b. Menambah atau mengurangi, suatu Informasi Elektronik dan/atau Dokumen Elektronik milik Orang lain atau milik publik.
   c. Merusak atau menghilangkan, suatu Informasi Elektronik dan/atau Dokumen Elektronik milik Orang lain atau milik publik.

5. Jenis Kejahatan : Hacker [Pasal 32 ayat 2]

   Ciri-Ciri:

   a. Memindahkan atau menyembunyikan suatu Informasi Elektronik dan/atau Dokumen Elektronik milik Orang lain atau milik publik
   b. Memindahkan Informasi Elektronik kepada Sistem Elektronik Orang lain yang tidak berhak.
   c. Mentransfer Dokumen Elektronik kepada Sistem Elektronik Orang lain yang tidak berhak.

6. Jenis Kejahatan : Hacker [Pasal 32 ayat 3]

   Ciri-Ciri:

   a. Terbukanya suatu Informasi Elektronik dan/atau Dokumen Elektronik yang bersifat rahasia menjadi dapat diakses oleh publik.
   b. Terbukanya suatu Informasi Elektronik dan/atau Dokumen Elektronik yang bersifat rahasia menjadi dapat diakses oleh publik dengan keutuhan data yang tidak sebagaimana mestinya.

c. Kejahatan Berat

1. Jenis Kejahatan :Pornografi [Pasal 27 ayat 1]

   Ciri-Ciri :

   a. Mempublikasikan video porno milik seseorang melalui sebuah situs web.
   b. Mengirim file ke email seseorang dengan unsur pornografi.
   c. Memanipulasi dan mempublikasikan gambar porno melalui media internet.
   d. Melakukan kejahatan pornografi

2. Jenis Kejahatan :Penyadapan [Pasal 31 ayat 1]

   Ciri-Ciri :

   a. Penyadapan akses komunikasi melalui video *conference* maupun chatting.
   b. Penyadapan data melalui saluran transmisi data (kabel telepon, serat optik atau satelit).
   c. Melakukan penyadapan

3. Jenis Kejahatan :Penyadapan [Pasal 31 ayat 2]

   Ciri-Ciri :

   a. Melakukan intersepsi atas transmisi informasi elektronik dan/atau dokumen elektronik yg tidak bersifat public dari, ke, dan di dalam suatu komputer milik orang lain.
   b. Penyadapan data melalui saluran transmisi data yang menyebabkan penghilangan atau penghentian data yg sedang di transmisikan.
   c. Melakukan penyadapan

4. Jenis Kejahatan :Pembajakan Software [Pasal 34]

   Ciri-Ciri :

   a. Membuat program kecil guna menerobos keamanan software tersebut untuk mendapatkan hak akses atas lisensi software.
   b. Menduplikasikan software milik pihak lain tanpa izin (ilegal).
   c. Menjual dan mendistribusikan software bajakan untuk mendapatkan keuntungan pribadi.

5. Jenis Kejahatan : Merusak jaringan [Pasal 33]

   Ciri-Ciri :

   a. Merusak sistem orang lain melalui jaringan internet tanpa sepengetahuan orang tersebut.
   b. Melakukan tindakan apapun yang berakibat terganggunya sistem atau jaringan elektronik.

6. Jenis Kejahatan :Pemalsuan Data [Pasal 35]

   Ciri-Ciri :

   a. Melakukan pemalsuan dokumen/ informasi milik perusahaan/instansi dengan cara memanipulasi data tersebut sehingga seolah-olah dianggap data yang otentik.
   b. Melakukan pengrusakan atau menghilangkan informasi/dokumen elektronik dengan tujan agar data tersebut dianggap seolah-olah data yang otentik.
   c. Melakukan perubahan atau penciptaan informasi/dokumen elektronik dengan tujuan data tersebut dianggap seolah-olah data yang otentik.

7. Jenis Kejahatan :Teror [Pasal 29]

   Ciri-Ciri :

   a. Mempublikasikan dokumen yang bersifat teror kepada pihak luas melalui email, situs web/ blog, dan melalui situs jejaring sosial.
   b. Mempublikasikan video yang berisi ancaman atau terror yang dapat meresahkan orang yang terlibat.
   c. Mengirim ancaman kekerasan dan menakut-menakuti yang ditujukan secara pribadi.

8. Pasal 36

   Ciri-ciri :

   Melakukan dengan sengaja perbuatan yang tergolong kedalam seluruh kejahatan ringan, sedang dan berat yang mengakibatkan kerugian orang lain.

2.6 Kajian Penelitian Terkait

Penelitian tentang sistem pakar untuk identifikasi kejahtan dunia maya belum terlalu banyak dikembangan. Dari beberapa penelitian yang ada, salah satu penelitian sistem pakar untuk identifikasi kejahtan dunia maya yang diteliti oleh Meilany pada tahun 2010. Pada penelitian ini sistem pakar tidak menggunakan metode apapun. Berdasarkan penelitian ini, sistem pakar hanya mampu memberikan kesimpulan pasal yang terkait dengan kejahatan berdasarkan Undang-Undang Informasi Transaksi Elektronik.

Sistem pakar untuk identifikasi kejahatan dunia maya juga pernah diteliti oleh Endah pada tahun yang sama, pada penelitian ini membahas tentang bagaimana sistem pakar dapat membantu mengidentifikasi kejahatan dunia maya berdasarkan ciri-ciri kasusnya sehingga dapat disimpulkan pasal apa yang menjeratnya.

Untuk metode VCIRS (*Variable-Centered Intelligent Rule System*) sendiri telah banyak digunakan dalam penelitian tentang sistem pakar, diantaranya Sistem Pakar Diagnosa Penyakit Paru-Paru Menggunakan *Variable- Centered Intelegent Rule Sistem* (Felvi, 2011). Pada penelitian ini sistem pakar di rancang untuk mendiagnosa penyakit paru-paru berdasarkan gejala yang dirasakan oleh pasien. Hasil akhir dari sistem ini adalah bagaimana setiap pasien yang melakukan diagnosa mengetahui penyakit paru-paru apa yang dideritanya.

Penelitian lainnya tentang metode VCIRS ini adalah Sistem Pakar Untuk Deteksi Dini Penyakit Menggunakan *Variable- Centered Intelegent Rule Sistem* (Arina, 2011). Pada penelitian ini sistem pakar juga dibuat untuk mendeteksi secara awal terhadap penyakit berdasarkan gejala-gejala yang dirasakan oleh pasien. Kesimpulan dari penelitian ini adalah bagaimana pasien mengetahui solusi dari setiap penyakit yang dirasakan.