RANCANG BANGUN COMPUTER-AIDED SOFTWARE ENGINEERING (CASE-TOOL) UNTUK SISTEM BERBASIS ATURAN MENGGUNAKAN DEPENDENCY DIAGRAM

Titik Lusiani[1)] **Oey She Khun** [2)]

1) Program Studi/Jurusan Manajemen Informatika, STIKOM Surabaya, email: lusiani@stikom.edu

2) Program Studi / Jurusan Sistem Informasi, STIKOM Surabaya

Abstract: CASE-Tool computer-based product supports software development process; uses for Rule-Based System is Decision Tree. Decision Tree method can process a tree as input and produce an Expert System Application. Knowledge acquisition process in Decision Tree method can produce massive and complex tree if there is a lot of information and categories to choose. Dependency Diagram could represent better knowledge acquisition for Expert System because it indicates the relationship among critical factors, input questions, rules, values, and recommendation made by the knowledge base system. CASE-Tool uses for Rule-Based System using Dependency Diagram to give better knowledge base for engineer to develop a Rule-Based System.

Keywords: CASE-Tool, Dependency Diagram, Rule-Based System, Expert System, Generate Rule

Sistem Berbasis Aturan merupakan suatu sistem pakar yang menggunakan aturan-aturan untuk menyajikan pengetahuannya. Dengan demikian Sistem Berbasis Aturan dapat dikatakan sebagai suatu perangkat lunak yang menyajikan keahlian pakar dalam bentuk aturan-aturan pada suatu domain tertentu untuk menyelesaikan suatu permasalahan. Contoh aplikasi Sistem Berbasis Aturan yang dikembangkan untuk menyelesaikan satu *domain* persoalan adalah Sistem Pakar untuk menentukan Interaksi Obat dengan menggunakan *Forward Chaining* yang diteliti oleh Sanjaya (2005).

Saat ini telah dikembangkan *Computer-Aided Software Engineering* (CASE) *Tools* untuk Sistem Pakar yang berperan sebagai *tool* untuk menghasilkan sebuah Sistem Berbasis Aturan dengan *multi domain*. CASE-*Tools* memberikan kebebasan bagi pengembang untuk membangun suatu *knowledge base* pada aplikasi, sehingga aplikasi-aplikasi Sistem Pakar yang dihasilkan tidak terbatas pada satu domain persoalan saja namun dapat meliputi banyak domain berdasarkan *knowledge base* yang dimasukkan.

Salah satu metode yang digunakan untuk membangun *knowledge base* Sistem Berbasis Aturan adalah *Decision Tree*. Metode *Decision Tree* dapat memproses masukan berupa *tree* dan menghasilkan sebuah aplikasi Sistem Pakar (Setio, 2005). Dalam implementasinya, metode *Decision Tree* memiliki kelemahan, yaitu proses perolehan pengetahuan (*knowledge acquisition*) dapat menghasilkan *tree* yang sangat besar dan kompleks apabila terdapat banyak informasi dan kategori yang harus dipilih.

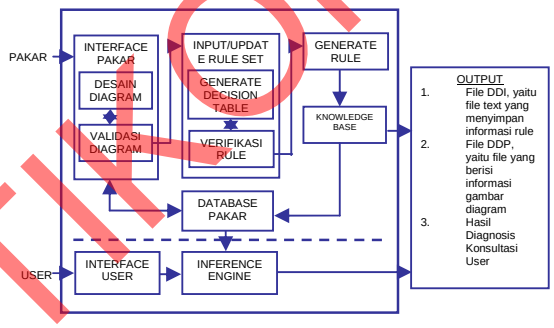
Berdasarkan persoalan tersebut, akan dibuat sebuah CASE-*Tool* untuk Sistem Berbasis Aturan dengan menggunakan *Dependency Diagram.* Keunggulan dari metode ini adalah penggunaan *Dependency Diagram* dapat mewakili *knowledge acquisition* Sistem Pakar yang lebih baik karena diagram ini menggambarkan hubungan yang jelas antara faktor-faktor kritis, pertanyaan-pertanyaan input, *rule-rule*, nilai-nilai, dan rekomendasi yang

dibuat. Tujuan dari penelitian adalah membuat CASE-*Tool* untuk Sistem Berbasis Aturan menggunakan *Dependency Diagram.* Dimana, sistem memproses masukan berupa *Dependency Diagram* dan menghasilkan keluaran berupa sebuah *rule-based system*. Dengan menggunakan *Dependency Diagram* diharapkan sistem dapat memberikan *knowledge base* yang lebih baik dan secara keseluruhan dapat membantu memudahkan *knowledge engineer* untuk mengembangkan suatu Sistem Berbasis Aturan.

METODE

Desain Arsitektur

Dalam proses pengembangan perangkat lunak, peneliti menggunakan sebuah desain arsitektur CASE yang menjelaskan elemen-elemen utama dalam sistem dan hubungan antar elemen-elemen tersebut untuk menghasilkan *output* berupa *rule-based system*. Desain arsitektur pengembangan secara umum ini dapat dilihat pada Gambar 1. Desain CASE yang dibuat secara garis besar terdiri dari dua desain utama, yaitu desain untuk pakar dan desain untuk user.



Gambar 1. Desain Arsitektur CASE secara umum

Berikut penjelasan dari desain arsitektur:

1. *Interface* Pakar: Media yang digunakan oleh pakar untuk mengembangkan sistem. Dimana dalam merancang sistem dengan membuat desain *Dependency Diagram* pada kanvas gambar. Interface Pakar terdiri dari dua proses utama, yaitu:
   a. Desain Diagram, merupakan proses pembuatan *Dependency Diagram* pada kanvas gambar.
   b. Validasi Diagram, merupakan proses pengecekan terhadap relasi dari tiap komponen dalam desain diagram.
2. *Input/Update Rule Set*: Proses penentuan parameter-parameter yang digunakan dalam tiap rule set. Proses ini meliputi dua subproses, yaitu:
   a. *Generate Decision Table*, ini merupakan proses untuk membentuk tabel keputusan (*Decision Table*) dari setiap *rule set* pada desain *Dependency Diagram*.
   b. Verifikasi *Rule*, merupakan proses pengecekan terhadap *rule-rule* yang terbentuk pada *Decision Table* apakah telah memenuhi konstrain-konstrain yang telah ditetapkan dalam sistem.
3. *Generate Rule*: Proses *Generate Rule* dijalankan untuk membentuk *rule* dalam tiap *rule set* berdasarkan *Decision Table* yang terbentuk.
4. *Database* Pakar: Digunakan untuk menampung sementara informasi *rule* pada *project* yang sedang dijalankan.
5. *Interface User*: Sebagai media oleh user untuk melihat dan berinteraksi dengan sistem pada saat proses konsultasi.
6. *Inference Engine*: Mekanisme inferensi yang digunakan adalah *Forward Chaining* karena sistem lebih dulu mengetahui fakta-fakta yang ada, kemudian mencari kesimpulan sementara sampai akhirnya berhenti setelah menghasilkan sebuah kesimpulan akhir.

Proses *Forward Chaining* diperlukan dalam mencari solusi berdasarkan *goal* konsultasi dan *rule base* yang ada dalam *working memory*.

7. *Knowledge Base*: Kumpulan fakta dan aturan (*rule*) serta *working memory* yang merupakan fakta yang diperoleh oleh sistem selama proses berlangsung.
8. *Output* :

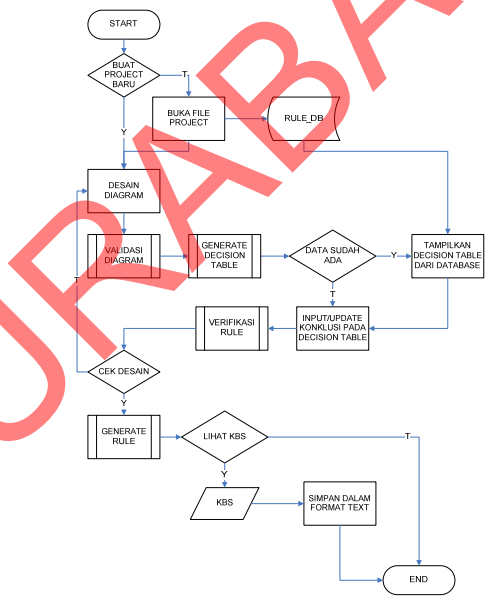
   a. *Output* dari desain pakar adalah file ddp (*dependency designer project*) yang berisi gambar *dependency diagram*, dan file ddi (*dependency designer info*) yang menyimpan informasi *rule* yang terbentuk.

   b. *Output* dari desain *user* adalah hasil akhir dari proses *inference* berupa laporan hasil konsultasi user.

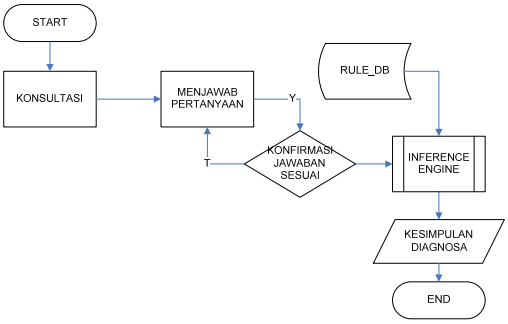
Perancangan Proses

Perancangan proses dalam sistem ini adalah sebagai berikut: 1)Diagram Alir Sistem untuk Desain Pakar, 2)Diagram Alir Sistem untuk Desain *User*, 3)Diagram Alir Sistem untuk Proses Validasi Diagram, 4)Diagram Alir Sistem untuk Proses *Generate Decision Table*, 5)Diagram Alir Sistem untuk Proses Verifikasi *Rule*, 6)Diagram Alir Sistem untuk Proses *Generate Rule*, 7)Diagram Alir Sistem untuk Proses *Inference Engine.*

Pada Gambar 2 menjelaskan desain diagram alir pakar untuk mendefinisikan aturan-aturan yang akan digunakan oleh Sistem Berbasis Aturan. Proses ini dimulai oleh pakar dengan membuka sebuah project atau membuat project baru. Selanjutnya, data-data yang berhubungan dengan informasi rule ini akan ditampung sementara ke dalam database. Pakar lalu merancang *Dependency Diagram* dengan cara menentukan parameter-parameter yang digunakan. Dari tiap *rule set* yang ada, selanjutnya pakar melakukan verifikasi rule. Kemudian pakar meng-*generate Decision Table* sehingga didapatkan *rule* untuk masing-masing *rule set.* Setelah itu desain secara keseluruhan akan diperiksa sebelum dapat melakukan *Generate Rule*. KBS yang dihasilkan dapat disimpan ke dalam format text terpisah.



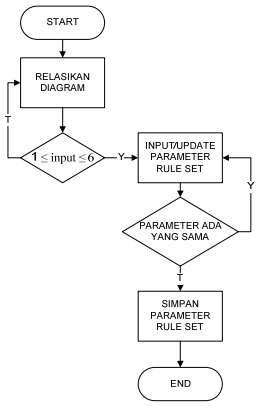
Gambar 2. Diagram Alir Sistem Desain Pakar



Gambar 3. Diagram Alir Sistem Desain *User*

Pada Gambar 3 menjelaskan proses jalannya sistem pada desain *user* pada saat konsultasi hingga mendapatkan kesimpulan/goal lewat proses inferensi. Melalui tahap konsultasi, user akan diajukan beberapa pertanyaan sesuai basis pengetahuan yang terbentuk. Dari jawaban-

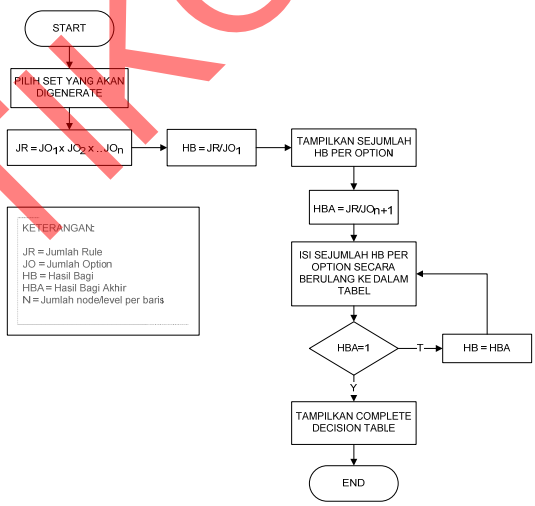
jawaban yang diberikan, sistem akan melakukan proses inferensi menggunakan metode *Forward Chaining* untuk mendapatkan kesimpulan akhir dari konsultasi ini.



Gambar 4. Diagram Alir Sistem Proses Validasi Diagram

Pada Gambar 4 menjelaskan proses Validasi sistem terhadap desain diagram yang dirancang oleh pakar. Proses Validasi ini meliputi pengecekan terhadap jumlah inputan dalam sebuah rule set dan pengecekan tiap parameter dalam rule set.

Hasil *Generate Decision Table* akan ditampilkan per set, sesuai dengan set-set yang terbentuk dari desain pakar. Diagram alir dari *Generate Decision Table* dapat dilihat pada Gambar 5.



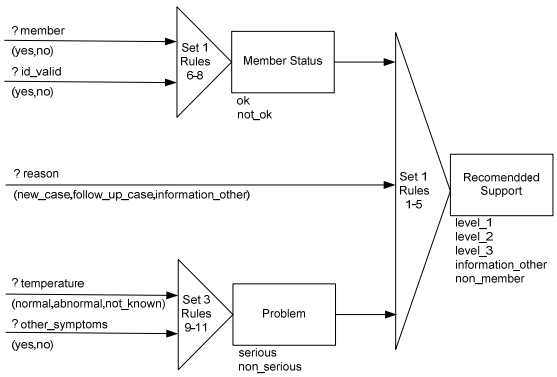
Gambar 5. Diagram Alir Sistem Proses *Generate Decision Table*

Proses *Generate Decision Table* merupakan proses pembentukan *Decision Table* secara lengkap untuk tiap *rule set*. Proses ini dimulai dengan menentukan *set* yang akan di-*generate* terlebih dahulu. Kemudian dicari jumlah *rule* (JR) yang akan terbentuk dari set tersebut dengan cara mengalikan jumlah *option* (JO) yang ada pada tiap *input*. Setelah didapat jumlah *rule* maka jumlah *rule* tersebut akan dibagi dengan jumlah *option* pada *input* pertama maka akan didapat nilai dari hasil bagi (HB), dimana hasil bagi ini merupakan jumlah per *option* yang akan ditampilkan. Untuk mendapatkan jumlah per *option* dari *input* selanjutnya maka HB harus dibagi dengan jumlah *option*-nya sehingga didapat nilai hasil bagi akhir (HBA). Langkah ini dilakukan hingga HBA mencapai nilai 1 yang berarti nilai HBA sama dengan jumlah *option* dari *input* akhir. Proses berhenti apabila semua *rule set* yang terbentuk dari desain pakar telah di-*generate*.

Proses-proses penting yang terdapat dalam *Generate Decision Table* adalah sebagai berikut:

A. Mendapatkan Jumlah Rule per Set

Jumlah rule-rule yang terbentuk dari proses *Generate Decision Table* didapat dari hasil perkalian *option* tiap-tiap *input* yang terdapat dalam suatu *rule set*. Sebagai gambaran dapat dilihat desain pakar untuk pelayanan kesehatan pada *Health Maintenance Organization* (HMO). Dalam hal ini, terdapat 3 kategori umum untuk menentukan dukungan kesehatan yang diberikan yaitu *member status*, *reason*, dan *problem* yang dapat dilihat pada Gambar 6.



Gambar 6. Desain dalam Bentuk *Dependency Diagram*

Pada Gambar 6 terdapat *rule set Member Status* yang terdiri dari 2 input yaitu *member* dan *id_valid*. Masing-masing *input* ini memiliki 2 *option*. Jadi untuk mendapatkan jumlah *rule* dari *set Member Status* adalah mengalikan jumlah *option* dari *member* dan *id_valid* (2 x 2) sehingga didapat 4 *rule*.

Untuk *rule set* yang memiliki inputan berupa *rule set* lain seperti *set Recommended Support* yang memiliki inputan berupa *rule set Member Status* dan *Problem* serta inputan *reason*. Maka untuk mendapatkan jumlah *rule* untuk *set Recommended Support* yaitu dengan mengalikan jumlah *option* dari tiap inputan (baik *rule set* maupun inputan biasa), sehingga didapatkan 2 x 3 x 2 = 12 *rule*. Jumlah *rule* total yang terbentuk dari *Dependency Diagram* ini adalah hasil penjumlahan dari jumlah *rule* tiap *rule set* yaitu: *Member Status*(4) + *Problem*(6) + *Recommended Support*(12) maka didapatkan sejumlah 22 *rule*.

B. Menampilkan Rule-rule Setelah Generate

Agar seluruh *node* yang ditampilkan dapat memberikan kombinasi dari semua premis maka jumlah *rule* dari *set* yang ditampilkan akan dibagi dengan jumlah *option* pada *input* pertama, maka akan didapatkan nilai dari hasil bagi (HB). Hasil bagi ini akan menjadi jumlah per *option* dari *input* yang akan ditampilkan. Untuk mendapatkan jumlah per *option* dari *input* yang akan ditampilkan selanjutnya, HB harus dibagi dengan jumlah *option* dari *input* selanjutnya sehingga didapatkan nilai hasil bagi akhir (HBA). Langkah ini dilakukan sampai HBA mencapai nilai 1 yang berarti nilai HBA sama dengan jumlah *option* dari *input* akhir.

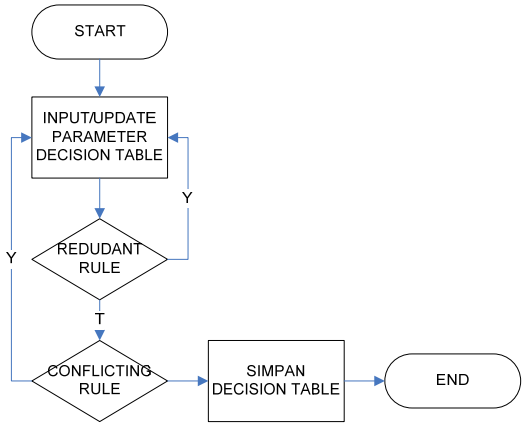
Sebagai gambaran dari kombinasi premis yang lengkap seperti dalam Tabel 1 yang menampilkan *rule-rule* yang terdapat dalam *set Recommended Support*.

Tabel 1. Contoh *Complete Decision Table* untuk *Set Recommended Support*

Rule	member	id_valid
1	yes	yes
2	yes	no
3	no	yes
4	no	no

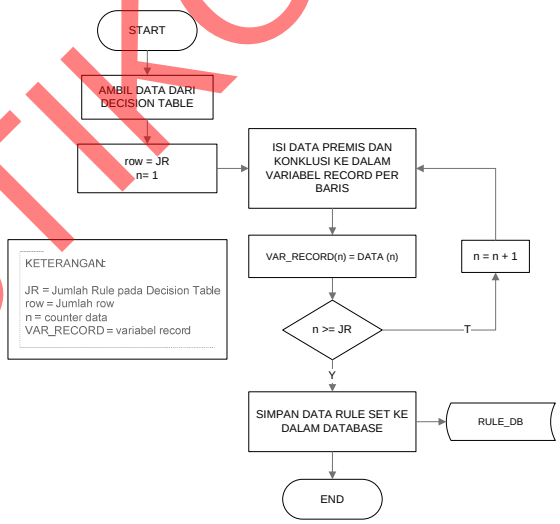
Terlihat pada Tabel 1 jumlah *rule* pada *rule set member status* adalah 4 *rule*. Maka jumlah *rule* tersebut dibagi dengan jumlah *option* premis pertama dari *member status* yaitu *member* sebanyak 2 *option*, sehingga dalam menampilkan tiap-tiap *option* dari premis *member* sebanyak 2 kali (4 *rule* dibagi 2 *option*). Hasil bagi dari jumlah *rule* dengan jumlah *option* premis pertama tadi akan dibagi kembali dengan jumlah *option* dari premis kedua yaitu *id_valid* yang memiliki 2 *option* sehingga tiap-tiap *option* dikenakan perulangan sebanyak 1 kali (2 dibagi 2 *option*). Apabila hasil bagi sudah mencapai angka satu maka proses akan dihentikan, karena ini berarti proses penampilan telah mencapai premis terakhir.

Dari *Decision Table* yang terbentuk dapat diperoleh serangkaian *rule* yang digunakan sebagai acuan proses inferensi *rule-based system*. Namun sebelumnya harus dilakukan proses verifikasi terhadap *rule-rule* yang terbentuk tersebut. Diagram Alir Sistem untuk proses Verifikasi *Rule* dapat dilihat pada Gambar 7.



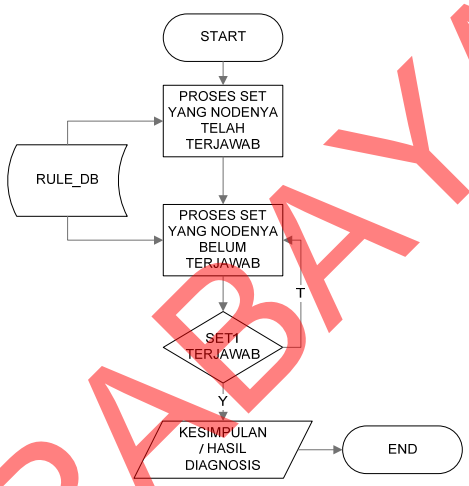
Gambar 7. Diagram Alir Proses Verifikasi Rule

Rule-rule ini didapatkan dari tiap baris data pada *Decision Table* dalam bentuk IF-THEN *statement* yang terdiri dari premis dan konklusi untuk tiap kondisi. Selanjutnya, *rule-rule* ini disimpan dalam *database*. *Rule-rule* tersebut terbentuk melalui proses *Generate Rule* yang dapat dilihat Diagram Alirnya pada Gambar 8.



Gambar 8. Diagram Alir Sistem Proses *Generate Rule*

Pada Gambar 9 menjelaskan proses *Inference Engine* dengan menggunakan metode *Forward Chaining* yaitu penalaran dari data-data yang ada untuk mencapai suatu konklusi.



Gambar 9. Diagram Alir Sistem Proses *Inference Engine*

Perancangan File

Agar Sistem Berbasis Aturan ini dapat menampung bermacam permasalahan dan bidang ilmu maka sistem dirancang untuk dapat menerima parameter-parameter ke dalam file. File-file yang digunakan adalah sebagai berikut:

1. *File Project*

   File ini digunakan untuk menyimpan *project rule-based system* yang dibangun. *File project* terdiri dari dua *file* utama, yaitu *file* ddp (*dependency designer project*) yang menyimpan desain *dependency diagram*, dan *file* ddi (*dependency designer info*) yang menyimpan informasi *rule* dari *project* yang bersangkutan. Untuk dapat membuka sebuah *project*, kedua *file* ini mutlak diperlukan.

2. Variabel *Record*

   Variabel ini berfungsi untuk menyimpan informasi *rule* sementara pada saat perancangan desain *Dependency Diagram* dan pada saat proses inferensi dijalankan.

3. *File* Hasil Diagnosis

   File ini digunakan sebagai hasil akhir dari proses konsultasi user pada Sistem Berbasis Aturan yang dihasilkan.

4. *File Database*

   Database digunakan sebagai media penyimpanan sementara pada saat proses perancangan diagram ataupun proses inferensi pakar. *Database* terdiri dari empat tabel yaitu tabel *Ruleset*, tabel *Rule*, tabel KBS, dan tabel KBS_Detail. Tabel *Ruleset* digunakan untuk menyimpan data *rule set* pada *Dependency Diagram*, sedangkan tabel *Rule* digunakan untuk menampung baris *rule* yang berhubungan dengan *rule set* tertentu. Tabel KBS dan KBS_detail digunakan pada saat proses inferensi sebagai basis pengetahuan untuk menyimpan semua informasi rule yang terbentuk pada proses *Generate Rule*. Struktur tabel yang digunakan dapat dilihat pada Tabel 2 (Tabel *Ruleset*), Tabel 3 (Tabel *Rule*), Tabel 4 (Tabel KBS), dan Tabel 5 (Tabel KBS_Detail).

Tabel 2. Tabel *RuleSet*

No.	Nama Field	Type	Lebar	PK	FK		Keterangan
					Tabel	Kolom	
1	rs_id	Number	Integer	PK			index ruleset
2	name	Text	32				label nama

Tabel 3. Tabel *Rule*

No.	Nama Field	Type	Lebar	PK	FK		Keterangan
					Tabel	Kolom	
1	rule_id	Number	Integer	PK	Ruleset	rs_id	index rule
2	input	Text	32				input rule
3	output	Text	32				output rule

Tabel 4. Tabel KBS

No.	Nama Field	Type	Lebar	PK	FK		Keterangan
					Tabel	Kolom	
1	rule_no	Number	Integer	PK			No. rule
2	Konklusi_var	Text	32				Variabel konklusi
3	Konklusi_val	Text	32				Nilai konklusi

Tabel 5. Tabel KBS_Detail

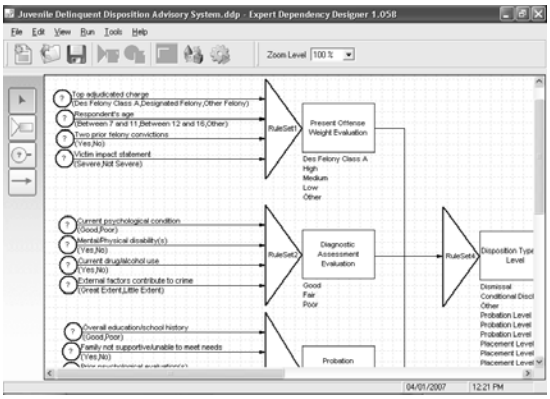
No.	Nama Field	Type	Lebar	PK	FK		Keterangan
					Tabel	Kolom	
1	idx	Number	Integer	PK	KBS	rule_no	Index detail rule
2	premis_var	Text	32				Variabel premis
3	premis_val	Text	32				Nilai premis
4	operator	Text	5				Op.perbandi-ngan

HASIL DAN PEMBAHASAN

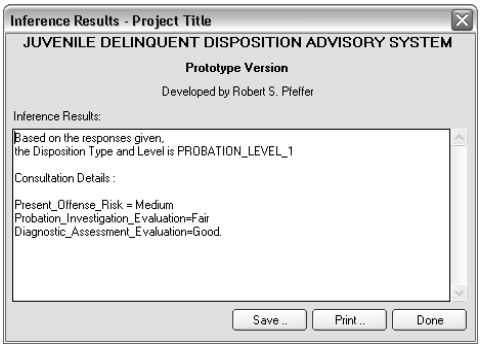
Pada tahap ini diuraikan hasil dan pembahasan penelitian terhadap perangkat lunak dimulai dari inputan ke proses penalaran.

Untuk pengujian logika dan hasil akhir pada Sistem Berbasis Aturan yang dihasilkan CASE-*Tool* menggunakan *Dependency Diagram* ini maka digunakan beberapa data yang bersifat pasti, dimana data-data tersebut diambil dari beberapa permasalahan yaitu: *Juvenile Delinquent Disposition Advisory System* dan *Student Financial Aid Advisor* dari buku *Developing Knowledge-Based Systems using VP-Expert* (Dologite,1993). Hal ini dilakukan agar dapat diketahui bahwa proses *generate rule* dapat menghasilkan *rule-rule* yang sesuai dan benar.

Berikut ini adalah contoh gambar hasil dari perancangan Sistem Berbasis Aturan untuk permasalahan *Juvenile Delinquent Disposition Advisory System*



Gambar 10. *Dependency Diagram* pada *Juvenile Delinquent Disposition Advisory System*



Gambar 11. Hasil Inferensi pada *Juvenile Delinquent Disposition Advisory System*

Sistem diuji cobakan untuk *User* dengan kondisi *Top Adjudicated Charge=Designated Felony, Respondet's Age=Between 12 and 16, Two prior felony convictions=no, Victim impact statement=not severe, Current psychlogical condition=good, Mental/physical disabilitiy(s)= no, Current drug/alcohol use=no, External factors contribute to crime=great extent, Overall education/school history=poor, Family not supportive/unable to meet needs=yes, Prior psychological evaluation(s)=good, Previous invlovement with drugs/alcohol=no, dan Prior social assistance=helpful.* Dari hasil inferensi didapatkan hasil diagnosis untuk *Disposition Type and Level* adalah *Probation Level 1* seperti pada Gambar 11. Berdasarkan uji coba di atas terlihat bahwa solusi yang dihasilkan oleh Sistem Berbasis Aturan yang dihasilkan oleh CASE-*Tool* menggunakan *Dependency Diagram* dari proses inferensi *forward chaining* telah sesuai dengan permasalahan yang telah di uji cobakan.

SIMPULAN

Beberapa kesimpulan yang didapatkan dari Sistem ini adalah: 1) CASE-*Tool* untuk Sistem Berbasis Aturan menggunakan *Dependency Diagram* ini dapat diimplementasikan pada permasalahan *Juvenile Delinquent Disposition Advisory System* menggunakan metode *forward chaining* dan menghasilkan rule-rule yang sesuai dan benar.Selain itu aplikasi juga dapat diimplementasikan ke berbagai permasalahan sistem pakar dengan metode *Forward Chainning* 2) *Rule-rule* yang dihasilkan dari proses *Generate Decision Table* merupakan kombinasi dari inputan-inputan dalam desain *Dependency Diagram*, dan telah melalui proses validasi diagram dan verifikasi *rule* sehingga kemungkinan terjadinya kesalahan pengisian dapat diperkecil. 3) Proses inferensi pada permasalahan *Student Financial Aid Advisor* dan *Juvenile Delinquent Disposition Advisory System* dapat menghasilkan konklusi yang sesuai dengan contoh kasus pada buku *Developing Knowledge-Based Systems using VP-Expert* (Dologite,1993).

DAFTAR RUJUKAN

Anonymous. 2004. *What Is a CASE Environment?*, (Online), (http://www.sei.cmu.edu/legacy/case/case_whatis.html, diakses 2 April 2006).

Anonymous, 2005, *Dependency Diagram*, (Online), (http://en.wikipedia.org/wiki/Dependency_diagram, diakses 2 April 2006).

Dologite, D.G. 1993. *Developing Knowledge-Based Systems using VP-Expert.* New York: Macmillan Publishing Company.

Durkin, John. 1994. *Expert Systems Design And Development.* New Jersey: Prentice Hall.

Gonzales, Avelino J and Dauglas, D Dankel.. 1993. *The Engineering Knowledge-Based System.* New Jersey: Prentice Hall.

Herwitanto, Riza. 2004. *Aplikasi Generating Rule untuk Sistem Berbasis Aturan.* Skripsi tidak diterbitkan. Surabaya: Program Sarjana Komputer STIKOM Surabaya.

Ignizio, James P. 1991. *Introduction to Expert System: The Development and Implementation of Rule-Based Expert System.* Singapore: McGraw-Hill, Inc.

Pressman, Roger S. 1992. *Software Engineering A Practitioner's Approach.* Singapore: McGraw-Hill, Inc.

Setio, Soeryata. 2006. *Computer Aided Software Engineering (CASE) untuk Sistem Pakar dengan menggunakan Metode Decision Tree.* Skripsi tidak diterbitkan. Surabaya: Program Sarjana Komputer STIKOM Surabaya.

Sutomo, Ario. 2005. *Pengenalan Computer Aided Software Engineering*, (Online), (http://www.sony-ak.com/articles/4/case_introduction.php, diakses 10 April 2006).