VISUAL JESS: AN Expandable Visual Generator of Oriented Object Expert systems

Amel Grissa-Touzi, Habib Ounally and Aissa Boulila

Abstract— The utility of expert system generators has been widely recognized in many applications. Several generators based on concept of the paradigm object, have been recently proposed. The generator of oriented object expert system (GSEOO) offers languages that are often complex and difficult to use. We propose in this paper an extension of the expert system generator, JESS, which permits a friendly use of this expert system. The new tool, called VISUAL JESS, bring two main improvements to JESS. The first improvement concerns the easiness of its utilization while giving back transparency to the syntax and semantic aspects of the JESS programming language. The second improvement permits an easy access and modification of the JESS knowledge basis. The implementation of VISUAL JESS is made so that it is extensible and portable.

Keywords— Generator of Systems Expert, Programming oriented object classifies, object, inheritance, polymorphism.

I. Introduction

THE development of an expert system (E.S) is often expensive. To decrease their development costs, one often makes recourse to the Expert System Generators (E.S.G) already developed [1],[2]. Several E.S.G have then been proposed in both literature and trade [1], [2],[3],[4],[5], [6],[7]. These generators propose a motor of inference with a knowledge representation method. Diagrams common general to all these generators include an interfacing for the expert in order to define the basis of knowledge, and an interfacing for the user to exploit the generated E.S [3],[4]. E.S generators might be distinguished by their fashion of reasoning, structures of knowledge representation, and the interfacing that are often based on a textual language [1],[2], [3], [4], [5], [6],[7]. Many generators are based on the logic of order 0 or order 0+ as synta. Others are based on the logic of order 1 or on the semantic networks [6],[8],[9],[10],[11],[12],[13].

Recently, the use of the paradigm objects as the basis of ES. A generator has become very popular. The most well known are the JESS [14], [15], [16] and CLIPS [17], [18] generators, which will be named GSEOO (expert system Generator oriented object) in the following. The main strength of the GSEOO is the utilization of concepts objects [19], [20], [21], which is very close to the perception of experts and users. This facilitated significantly the writing of the knowledge basis (BC). Languages of definition of this BC use the concept of frame [19]. This fashion of representation, of hierarchical nature, is very complex to understand notably by users no insiders. Besides, the expert must master the language of definition that varies from a generator to the other. On the other site, in our knowledge, there is no GSEOO which offers an interfacing that gives back transparent concepts of representation and the underlying conventions to the specific language of the generator.

We propose in this article an extension of the JESS, in order to give back its comfortably usable even by users no insiders. This tool, called VISUAL JESS, bring two main improvements to JESS. The first improvement concerns the easiness of its utilization while giving back transparency to the syntactic and semantic aspects of the JESS programming language. The second improvement permits an easy access and modification of the JESS knowledge basis. The implementation of VISUAL JESS is made so that it is extensible and portable.

The paper is organized in four sections. The main characteristics of JESS, its strength and limits are presented in Section 1. The extension of JESS, called VISUAL JESS that we propose is described in Section 2. Section 3 present the main choices done in the implementation of VISUAL JESS while putting the accent on the extensibility. The balance of this work and its future perspectives are discussed in Section 4.

II. THE GSEOO JESS

The ES are widely used in various applications [1],[5],[8]. To decrease costs of their development, one often makes recourse to the Expert system Generators (GSE) developed already [1],[2]. Generators essentially include a motor of inference, a language of expression of knowledge and structures and conventions of representation. [3],[4]

Several generators have been proposed in the literature and in the trade. Some are even very old as EMYCIN [13]. The representation of the knowledge and the fashion of reasoning vary significantly from one generator to the other. Sinta is based on the logic of propositions [10], SNARK uses the logic of order 1 [6]. Lately, with the success of concepts of the

Manuscript received March 30, 2005.

A Grissa-Touzi., A. Boulila is with the Department of Electrical Engineering of National School of Engineers of Tunis, Tunisia (e-mail: amel.touzi@ enit.rnu.tn, aissa.boulila@laposte.net).

H Ounally. is Faculty of Sciences of Tunis, Department of Computer Science of Tunis, Tunisia (e-mail: Habib.ounelli@fst.rnu.tn).

paradigm object [19],[20],[21], several GSES adopted this paradigm as CLIPS[17,18] and JESS[14],[15],[16]. In this paper, a special focus has been assigned to the generator oriented JESS object (Java Ex-pert System Shell) of which we propose an extension that provides two major improvements.

Before presenting this extension, we are going to present the main characteristics of JESS and discuss the difficulties met in its utilization.

Our choice of JESS is justified by several reasons. The fashion of reasoning of JESS bases itself on the Rete algorithm that very efficient [15]. It is free software (open source) and this facilitated its integration in VISUAL JESS. JESS offers one API in very rich java. It supports the approach object in its fashion of reason-lies and in the representation of knowledge. This homogeneity makes it more interactive and expandable in comparison with other generators of the same nature.

JESS uses the notion of frames [8],[19] for the representation of knowledge. A frame is a hierarchical representation that includes several components (slots, facet, datum and how). Each frame definition is made with the codes deftemplate that must specify all components.

((deftemplate <deftemplate-name> [extends <classname>] [<doc-comment>]

[(slot <slot-name> [(default | default-dynamic <value>)][(type <typespec>))]*)

Example

(deftemplate personne extends être-vivant (slot nom (type STRING))(slot sexe (type STRING)(default homme))

The overlapping that can go multilevel in slotses and their facets give back the definition of a complex and with difficulty comprehensible frame.

This syntactic complexity of the JESS language requires from the expert a significant effort to end to write its basis of knowledge correctly (B.C) (its rules). Otherwise, the definition of the B.C makes himself from the line of command. Concepts of the paradigm object, notably the class concepts, inheritance and polymorphism, used by JESS, are not discerned explicitly by the expert of autat more that these concepts are not still assimilated well. So JESS gives back even difficult the stains the expert. Notice that the JESS language includes about fifty words spare.

To summarize, the expert who wants to use JES, must master the paradigm object and the hierarchical notion of frame on which are based the language of JESS.

For all these reasons, it seems natural that an extension of JESS that gives back transparency to its language can make it easily exploitable notably by the no experts. This extension can also be applied to other GSEOOS.

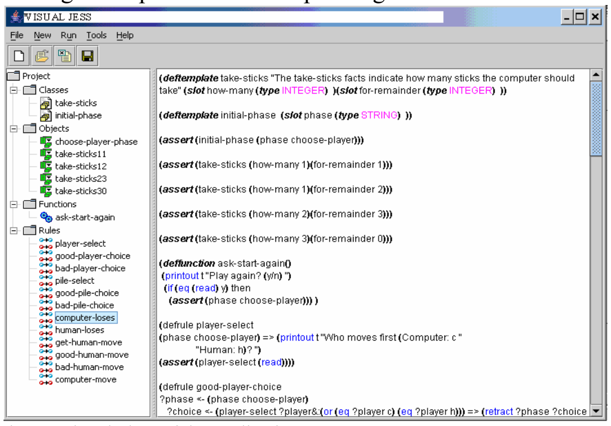
III. PRESENTATION OF VISUAL JESS

We propose in this paper a tool that spreads JESS, called VISUAL JESS in order to facilitate the generation of an ES based on the OO concept for users in individuals those insiders. As JESS, VISUAL JESS is developed in Java.

Indeed VISUAL JESS offers a convivial interfacing, which on the one hand present explicitly concepts objects and on the other hand encapsulate the JESS language.

The principle of VISUAL JESS is to allow an expert to define his classes, his objects, his functions and his rules of graphic manner and generate automatic-lies the corresponding frames in accordance with the language of JESS.

VISUAL present JESS to the expert a main screen (face 1) similar to the stan-sting usually used in the IDES (Integrated Development Surround-lies). The main screen essentially includes two parts. The left part is the navigator of components of basis of the B.C (classes, objects, functions, rules). The right part shows the corresponding script generated automatically by JESS. The navigator of classes permits selecting a component and manipulating its elements.



```
(deftemplate take-sticks "The take-sticks facts indicate how many sticks the computer should
take" (slot how-many (type INTEGER) )(slot for-remainder (type INTEGER) ))

(deftemplate initial-phase  (slot phase (type STRING) ))

(assert (initial-phase (phase choose-player)))

(assert (take-sticks (how-many 1)(for-remainder 1)))

(assert (take-sticks (how-many 1)(for-remainder 2)))

(assert (take-sticks (how-many 2)(for-remainder 3)))

(assert (take-sticks (how-many 3)(for-remainder 0)))

(deffunction ask-start-again()
 (printout t "Play again? (y/n) ")
  (if (eq (read) y) then
   (assert (phase choose-player))) )

(defrule player-select
(phase choose-player) => (printout t "Who moves first (Computer: c "
             "Human: h)? ")
(assert (player-select (read))))

(defrule good-player-choice
?phase <- (phase choose-player)
  ?choice <- (player-select ?player&:(or (eq ?player c) (eq ?player h))) => (retract ?phase ?choice
```

Fig. 1 main window of the application

We retail the different components of VISUAL JESS below

A. Component Classes

A class is identified by its name. Every class contains a whole of attributes having each by default a name, a type, a commentary and a value. A control of conformity between the value and the type of the attribute is done automatically. At the time of the safeguard, the user can see the JESS script generated for the inserted class.

A class can have a related class of which it inherits all attributes. A control on the cyclic inheritance is launched. A class, which is already inherited or instanced, cannot be modified. Figure 2 presents the screen of class seizure.

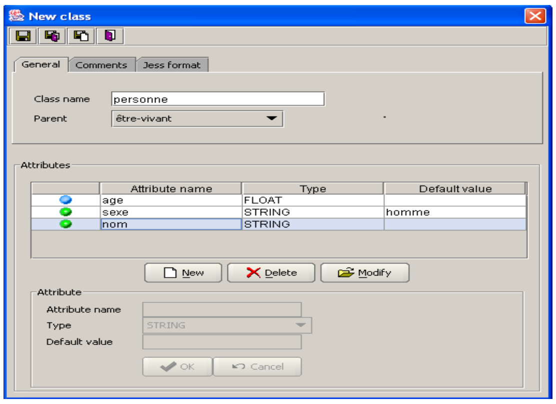
Example:

Classe « personne »

attribut «nom» : chaîne de caractère

attribut « sexe » : chaîne de caractère, par défaut homme

hérite « être-vivant »



Class name: personne
Parent: être-vivant

Attributes

	Attribute name	Type	Default value
🔵	age	FLOAT	
🟢	sexe	STRING	homme
🟢	nom	STRING	

Attribute name:
Type: STRING
Default value:

Fig. 2 screen of Seizure of a Class

VISUAL JESS transforms this menu in the following script:

(deftemplate personne extends être-vivant (slot nom (type STRING))(slot sexe (type STRING)(default homme))

Notice that the aged attribute has the blue color since it is inherited of the related class.

B. *Component Objects*

An object is identified by its name. In an object, attributes of a class are valued. A Boolean field permits to select this object, which will be used in the initialization of the knowledge basis.

To define an object of the class, the user must specify the name of the object, class instanced, values of attributes of class instanced and classes mothers. He must specify in addition if this object will be present in the initial BC.

Example

Object : MED

nom = Mohamed

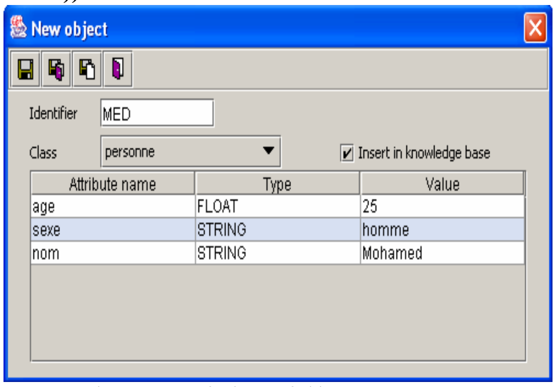
age = 25

Sexe = homme

The present face 3 the screen of seizure of an object.

VISUAL JESS transforms this menu in the following script:

(assert personne (nom « Mohamed ») (age 25) (sexe homme))



Identifier: MED

Class: personne

☑ Insert in knowledge base

Attribute name	Type	Value
age	FLOAT	25
sexe	STRING	homme
nom	STRING	Mohamed

Fig. 3 Screen of seizure of object

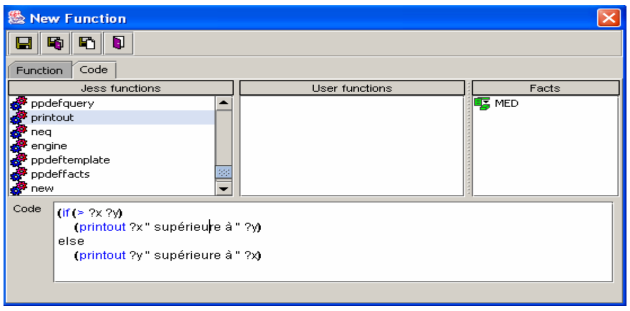
C. *Component Functions*

A function is identified by a name. The function can either have a set of variables on which it can act. The user can insert in the code its functions, functions created previously, of objects or the JESS functions. Figure 4 presents the screen of function seizure.

To define a function, the user must specify the name of the function, its parameters and the treatment to make.

Example:

Definition of the function plus-grand(x,y) that displays " x est supérieure à y " so x is bigger than there and " y est supérieure à x " otherwise.



Jess functions	User functions	Facts
ppdefquery		MED
printout		
neq		
engine		
ppdeftemplate		
ppdeffacts		
new		

Code

```
(if (> ?x ?y)
   (printout ?x " supérieure à " ?y)
else
   (printout ?y " supérieure à " ?x)
```

Fig. 4 screen of seizure of a function

VISUAL JESS transforms this menu in the following script:

(deffunction plus-grand(?x,?y)

(if (> ?x ?y)) (printout(?x « supérieure à » ?y)))

else (printout (?y « supérieure à » ?x)))

D. *Screen of rules*

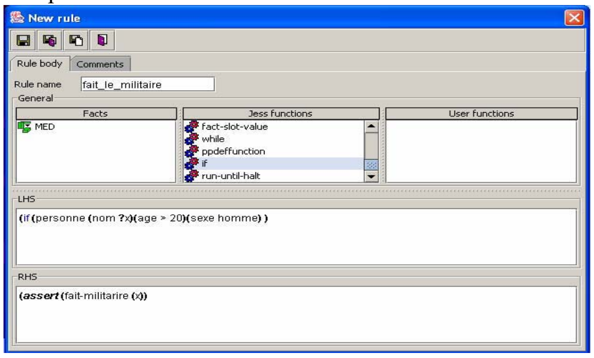
A rule is identified by a name. The rule possesses a Left Hand Side and a Right Hand Side. Like for the function the user can insert functions, which were created previously, objects or the JESSS functions.

To define a rule of the BC, the user must specify the name of the rule, the list of premises of the rule and his conclusion.

Example:

Une personne fait le militaire si : sexe masculin et age > 20

The present face 5 the screen of seizure of a rule.



Rule name: fait_le_militaire

Facts	Jess functions	User functions
MED	fact-slot-value	
	while	
	ppdeffunction	
	if	
	run-until-halt	

LHS

```
(if (personne (nom ?x)(age > 20)(sexe homme) )
```

RHS

```
(assert (fait-militarire (x))
```

Fig. 5 screen of seizure of a rule

VISUAL JESS transforms this menu in the following script:

(defrule fait_le_militaire (if (personne (nom ?x) (age > 20)(sexe homme)) (assert (fait-militarire (x)))

IV. Implementation of VISUAL JESS

Since the JESS generator is implemented in java, VISUAL JESS has been developed in Java [22],[23]. We give in this section, the main choices done in the implementation of VISUAL JESS.

For every element to manage a java class has been conceived, which contain methods and the necessary attributes to the management of this entity. Table 1 presents an illustration of the correspondence between the JESSS entities and classes conceived in this application:

TABLE I
Correspondence between the JESSS entities and the classes VISUAL JESSS

JESS	Definite JESS Classes for VISAUL JESS under JAVA
Template	Template
Fact	Template Instance
Slot	Attribute
Rule	Rule
Function	Function

To facilitate the extension and the portability of VISUAL JESS the implementation has been developed in five packages:

- Package jess manages the interaction with the user.
- The jess.kb package contains classes that implement the basis of knowledge
- The jess.resouces package contains the requested resources files for the application under study
- The jess.images package contains pictures used by the application.
- The jess.base package contains the utilitarian of basis of the application.

The jess.resouces package contains the syntactic details of the JESS language. It is the only package that it is necessary to modify to adapt VISUAL JESS to another GSEOO. This modification consists in replacing a file of syntax by another. This approach permits a complete extensibility of VISUAL JESS.

V. Conclusion

The GSEOOS has been very popular in the industrial world. Their utilizations are not still an easy task even for user's insiders. Indeed, it is necessary to often master a language of definition of the BC and concepts used in the representation of knowledge like frames.

We proposed in this paper an extension of JESS expert system generators, called VISUAL JESS that brings two main improvements. The first improvement encapsulates the syntactic details of the JESS language that are relatively complex since it is based on the hierarchical structure of frame. The second improvement gives back concepts of the paradigm object clarify and direct-lies usable by the expert. The implementation of VISUAL JESS is made so that it can be adapted to all other GSEOO giving back easy thus its extensibility to other GSEOO as well as its portability on others flat shapes. A first version is operational under windows. To our knowledge such a tool was not again proposed.

As perspective future we essentially mention this work:

– The possibility to have attributes of type object in the definition of a class.
– The generation of an expert system while leaving directly more from a level abs-feature like example a diagram of UML classes.
– The definition of a debugging tool of the generated HIMSELF to the use of the expert hu-hand.

References

[1] P. Frot, "Trois systèmes experts en Turbo pascal," sybex, 1987.
[2] J.P. Delahaye, : "Système expert : organisation et programmation des bases de connaissance en calcul propositionnel," Eyrolles, Paris, 1987.
[3] H. Farreny and M. Ghallab, "éléments d'intelligence artificielle. Hermes," 1987.
[4] F. Denis, "Cours d'intelligence artificielle - Systèmes experts ," http://www.grappa.univ-lille3.fr/polys/se/index.html, 1995.
[5] F. Henri, "Les systèmes experts : principes et exemples," Cepadues editions, 1989.
[6] A. Bonnet, "L'intelligence artificielle Promesses et Réalités," interEdition ,Paris, 1984.
[7] J.L. Laurière, "Intelligence artificielle: résolution de problème par l'homme et la machine," Editions Eyrolles, Paris 1986.
[8] A. Lescort, "Intelligence artificielle et systèmes expert," cedic 1985, Paris
[9] "Expert SINTA manuel d'utilisation," laboratoire de l'intelligence artificielle, http://www/lia.ufc.br.
[10] A. Adjaoute, "Rylm Générateur de systèmes experts pour la résolution de problèmes diagnostiques," http://scia.epita.net/files/common/others /RYLM.pdf.
[11] J.L. Michel, "Les systèmes expert, Turbo expert pour experts pressés," http://perso.wanadoo.fr/jean.luc.michel/Articles.jlm/Turbo.Expert.pdf.
[12] J. Lebbe and R.Vignes,"Utilitaires Xper," http://lis.snv.jussieu.fr/apps/xper/doc/utilxper/, 2003.
[13] W. Van MELLE, "A domain-independant production-rule system for consultation programs," VIth, IJCAI, Tokyo, pp 923-925,Aug.1979.
[14] J.F. Ernest,"Jess,The Rule Engine for the java plateform ," mai 2004, http://herzberg.ca.sandia.gov/jess.
[15] C. Lachaize, "Hamap Expert Rules Based System," Janvier 2003, INRIA Rhône-alpes, http://www.inrialpes.fr/helix/SIB/sibelius.html.
[16] R. Charton, "Jess, un outil pour générer des systèmes experts," équipe Maia-Loria, site web : http://perso.wanadoo.fr/chrtonrs/docs/serp.ppt.
[17] J.L. Crowley, "Système de productions ; CLIPS 6.0," Novembre 2000, http://www-prima.imag.fr/prima/Homepages/jlc/courses/2004/ ensi2.SIRR/Ensi2.SIRR.s5.pdf.
[18] J.C.Giarratano, "CLIPS, User's guide,version 6.20 ," March 2002, http://www.ghg.net/clips/download/documentation/usrguide.pdf.
[19] G. Masini, A. Napoli, D. Colnet, D. Léonard and K. Tombre, "les langages à objets, Langage de classes, langage de frames, langage d'acteurs," interEditions, Paris 1989.
[20] B. Meyer, "Conception et programmation orientées objet," ed Eyrolles, 2000.
[21] B. Meyer, "Conception et programmation par objet pour du logiciel de qualité," InterEditiond, Paris, 1990.
[22] P. Chan, "Le dictionnaire officiel JAVA2," ed Eyrolles, 1999.
[23] J.B. Boichat,"Apprendre Java et C++ en parallèle," Eyrolles, 2000.