New Approach for Conception and Implementation of Object Oriented Expert System Using UML

Amel Touzi and Mohamed Ben Messaoud
National School of Engineer of Tunis, Tunisia

Abstract: *Since conception is the primary part in the realization of a computer system and in order to help designers describe their software, several languages and tools such the UML modelling language have been proposed in the literature. UML knew an important success for the conception of object oriented systems. In this paper, we propose a new approach of conception and implementation of object oriented expert system based on the UML. For this we introduce our approach of design of the object oriented expert system based on UML, then we define an extension of the CLIPS, called VCLIPS_UML, in order to support UML. VCLIPS_UML brings two main improvements to CLIPS. The first improvement permits an easy access and modification of the CLIPS knowledge base. The user introduces his knowledge base described with the UML class and the object diagram; VCLIPS_UML gives the corresponding script directly. The second improvement concerns the ease of its utilization by making the syntactic and semantics aspects of the CLIPS programming language more transparent. The implementation of VCLIPS_UML is carried out in a way to make it expandable and portable.*

Keywords: *Expert system, expert system shell, UML, object oriented programming and GUI.*

Received March 16, 2007; accepted June 15, 2007

1. Introduction

Today, when we wishes to achieve a software system, it is necessary to represent, to specify, to construct and to document this system beforehand with the help of an adapted language admitted by the software specialists community. With the growing success of the object oriented programming, numerous semi-formal analyses and conception methods have been dedicated to this new paradigm. Today, the Unified Modeling Language (UML) is the most used language; it tends to become a standard [17]. It permits to describe the plans of construction of the software system, by integrating all the conceptual elements, all functions and all databases, all classes and all software components. It also permits to validate these plans before beginning the programming and to achieve the system [6, 17, 18].

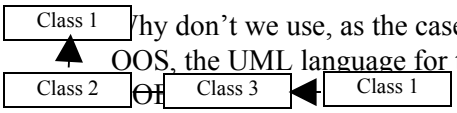
Effectively, the success of the object paradigm [16] led to the appearance on the market of several Object Oriented Systems (OOS) in various fields, of different programming languages, of various ES [14, 15] and ES Shell (ESS) of built ES using this paradigm. As example of shell, we can mention CLIPS [9] and JESS [7, 8]. The interest of this ESS is the use of the object concepts [7, 8, 9, 14, 15, 16] in the definition of the Knowledge Base (KB) of the generated ES.

In spite of the big success of UML and its use in varied domains [3, 19, 21], a theoretical study permitted us to raise the following points:

- Few work tried to use UML in the domain of the AI or the ES. We mention like example: berardi in [4, 5] which proposed a method of reasoning on UML class diagram and bauer in [2] which used UML for the system multi-agent.
- The mode of hierarchical representation, offered by Object Oriented Expert System (OOES) Shells (OOESS) is too complex to assimilate particularly for non initiated users. Defining of the KB described with the OO concept is not an easy task. Besides, to create an ES, the expert must master the language of definition that varies from a generator to other. To remedy this problem, several researchers proposed visual versions for this ESS. We mention like example JessGUI [13], JavaDON [20] or visual JESS [11] for JESS.

To our knowledge, an approach of design of the ES using UML was not still defined and there is not yet OOESS which support language UML. Hence following questions are worth asking:

- Why don't we define the ESS offering an interface that makes the syntactic and semantic aspects of the ESS specific language transparent?



Why don't we use, as the case of the different OOS, the UML language for the design of the OI

- Why don't we allow the user to describe his KB using UML language and to have ESS support this language?

In this paper, we present our contribution to answer to these questions: we introduce our approach of design of the OOES based on UML, and then we define an extension of the CLIPS, called VCLIPS_UML, in order to support the UML. This ESS brings two main improvements to CLIPS:

- To support the UML class and the object diagrams: the designer of system describes his KB with a UML class diagram; VCLIPS_UML generates the corresponding script automatically.
- To make it easily usable even by non initiated users. It's possible by making transparent the syntactic and semantic aspects of the CLIPS programming language.

Besides this introduction, this paper is organized in six sections. In section 2, we present the UML language and CLIPS shell. Section 3 presents problems and contributions. The design of the OOES with UML is presented in section 4. The extension of CLIPS, called VCLIPS_UML that we propose is described in section 5. Section 6 present the main choices done in the implementation of VCLIPS_UML while putting the accent on the extensibility. The balance of this work and its future perspectives are discussed in section 7.

2. Background

2.1. The UML Language

The UML language is a very rich graphic and textual language. UML Notation is divided into various types of diagrams, which describe complementary but not disjoined aspects of the modeled system. To instance we mention the following diagrams: (1) Case diagrams that describe the system functionality from the user's point of view, (2) class diagrams that describe the structure and substructure of the system using objects, attributes, operations, and associations. (3) object diagrams that illustrate the class diagram. In this paper, we will particularly describe the **c**lass diagram. For more details on the other diagrams, refer to [3, 6, 17, 18].

The class diagram in Figure 1 models the static structure of a system, in term of classes and relations between these classes. It is described by a collection of elements modelling structure by the system making abstraction of the dynamic and temporal aspects. It is the essential axis of the modelling object as well as the richest notation of all UML diagrams.

Figure 1. Class diagram.

A class is characterized by its name, the list of its attributes and the list of its operations (analogous to the methods). UML defines three levels of visibility of a private and protected public element we prefixed the element respectively by the symbol +, - #. UML defines four main types of relation between classes: (1) the association (denoted by), (2) the aggregation (denoted by), (3) the composition (denoted by) and (4) the generalization (denoted by). We can specialize a super-class, by adding to the subclass of the attributes and operations It is said that the super-class is a generalization of the subclass.

The big success of UML is due to several points: it is open and its visual aspect facilitates the comparison and the assessment of solutions. It permits to master the technical complexity thanks to the formalism and to the abstraction provided by the UML concepts.

2.2. The ESS CLIPS

To decrease costs of their development, we often resort to the already ESS developed [10, 12]. Shells essentially include an inference engine, a language of expression of knowledge and structures and conventions of representation [10, 12].

Recently, with the success of concepts of the object paradigm [16], several ESS adopted this paradigm OOESS as JESS [7, 8] and CLIPS [9] by adding a layer COOL which supports the object concept. Unfortunately, these ESS offers languages that are often complex and difficult to use.

In this paper, we are interested in the shell CLIPS of which we propose an extension. Before presenting this extension, we will show the principal characteristics of CLIPS and will show the difficulties encountered in its use and specially at the COOL level of layer. The CLIPS language permits to describe three types of facts:

- Ordered facts: they permit to represent the simple knowledge using the pattern concept.
- Structured facts: they permit to represent the structured knowledge. They are represented by templates.
- Layer CLIPS Object Oriented Language (COOL): it permits to represent the object oriented concept.

The COOL layer is a sort of hierarchical templates with multiple inheritances; it includes the object oriented concepts as: encapsulation, dynamic

generation of the instance, inheritance of the values and procedures in a hierarchy of objects, communication by message and polymorphism. A class of objects is defined by: a name, her parents (super-classes), her Sons (the sub-classes and instances), a list of attributes "slots" or property) with their description "facets" and a list of procedures (methods). All the classes programmed by the programmer must inherit directly or indirectly from the abstract class USER, a control on the cyclic inheritance is launched. A class that is already instanced or inherited cannot be modified.

Clips use the following syntax for the definition of a class*:*

(defclass <name> [<comment>] (is-a <superclass-name>+) [<role>] [<pattern-match-role>] <slot>* <handler-documentation>*)__
It use the following syntax for the definition of an object of a class:
(make-instance <instance-definition>) (active-make-instance <instance-definition>) <instance-definition> ::= [<instance-name-expression>] of <class-name-expression> <slot-override>* <slot-override> ::= (<slot-name-expression> <expression>*)

Example:

- To create the class person one described by the attributes name and age. In language CLIPS it will be written as follows:

  (defclass PERSON (is-a USER) (role concrete) (pattern-match reactive) (slot Name (type STRING) (default "Salah") (storage shared) (propagation inherit) (visibility public)) (slot Age (type INTEGER) (default 25) (create-accessor read-write) (access read-write) (storage shared) (propagation inherit) (source composite) (visibility public) (pattern-match reactive)))

- To create an instance of this class, the PERSON having the name "Sonia" and Age "20". In language CLIPS it will be written like follows: *(make-instance pers-1 of PERSON (Name "Sonia") (Age 20))*

Conclusion:

Through this example, we can show the difficulties encountered in its use and especially at the COOL level of layer. This complexity of syntax of the CLIPS language requires, from the expert, a significant effort in order to write correctly its KB. Moreover, the definition of the KB is done starting from the line of command. The concepts of the object paradigm, notably the class concepts, of inheritance and polymorphism, used by CLIPS, are not discerned explicitly by the expert especially as these concepts are not always well assimilated. Thus, CLIPS make the stains of the expert even more difficult. Notice that the CLIPS language includes about fifty words spare.

3. Problems and Contributions

The main strength of the ESS is the utilization of concepts objects [11, 12], which is very close to the perception of experts and users. This facilitated significantly the writing of the KB. Languages of definition of this KB use the concept of frame [11]. 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. Indeed, the expert who wants to employ CLIPS or JESS must master their proper language.

In addition the designers of OOS are accustomed to working with the UML. So, the question which arises then: why not introduce this concept in ESS? Certainly, for an OOES designer, the ideal is to have an ESS which supports OO concepts and UML language.

For all these reasons, it seems natural that an extension of CLIPS which makes its language relatively transparent can only make it easily exploitable particularly by no experts and support UML.

4. New Approach for Designing OOES Using UML

As in any computer application, we can define the various diagrams of UML for an ES. In this section, we limit to the UML class and the object diagram, to model the KB of an ES, which allows modelling the static view of an ES.

4.1. **Basic Concepts**

An ES is called OOES if its KB can be described with the object concepts. The KB of ES is made up of a Facts Base (FB), containing all the known facts of the system relating to a particular problem actually treated and of Rules Base (RB) made up of a list of rules containing the know-how of the expert expressed as rules.

We present in Table 1 the mapping between KB of ES and UML.

Table 1. Mapping between KB of ES and UML.

Knowledge Base of ES	UML
FB	Object Diagram
RB	Class Diagram

Example:

Our goal is to conceive a KB for a classification of vehicles. The class diagram relative to the description is present in Figure 2.

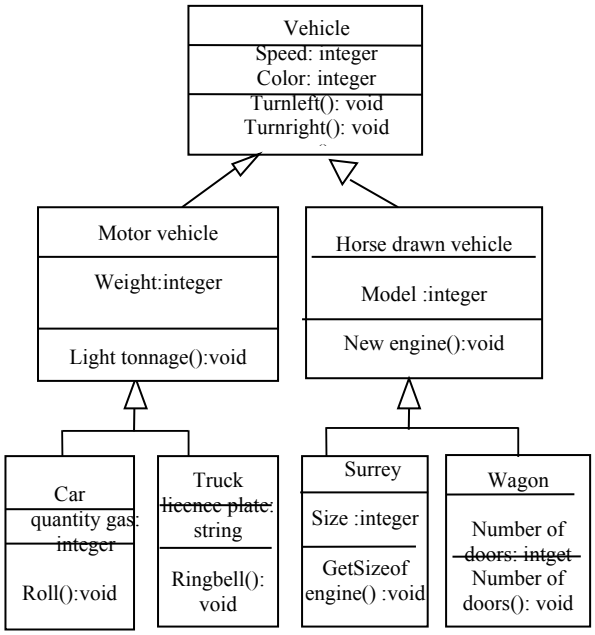


Figure 2. Class diagram for a classification of vehicles.

4.2. **Correspondence Between UML Diagram and an ESS**

As we presented above, to study the correspondence between UML and an ESS returns to study the correspondence between the UML class and the object diagrams, presented the KB and FB of ES respectively, and an ESS. To study this correspondence, we chose CLIPS like ESS. Our choice is justified by several reasons. It is free software. It supports the approach object in its mode of reasoning and in the representation of knowledge. This homogeneity makes it more interactive and extensible than other similar generators. The mapping between UML class diagram and CLIPS language is represented in Table 2. The mapping between UML object diagram and CLIPS language is represented in Table 3.

Table 2. Correspondence between UML class diagram and clips.

UML Class Diagram	Class of CLIPS
Class	Defclass
Attributes Name Type attributes Right of access	Slot Name Type Visibility
Simple inheritance	Propag : inherit|no-inherit (is-a name of class)
Multiple inheritance	Propag: inherit|no-inherit (is-a name of class1 name of class2)
Methodes Signatures (Void name of method (int x)) Right of access	Defmessage-handler Signatures (name The value of the return of a method is the result of the assessment of the last action in the method.) Primary, after, around

Table 3. Correspondence between UML object diagram and CLIPS.

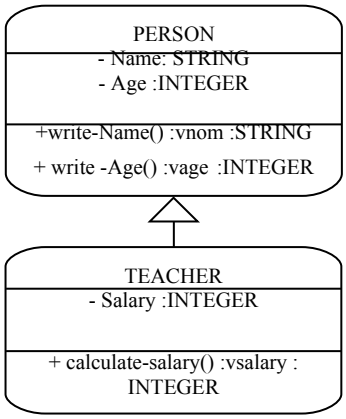
UML Object Diagram	Object in CLIPS
Object	Objet : make instance
Attributes Name Type of attributs Right of access	Slot Name Type Visibility
Simple inheritance	Propag : inherit|no-inherit (is-a name of class)
Multiple inheritance	Propag: inherit|no-inherit (is-a name of class1 name of class2)

Example:

- Either to create the class “PERSON” described by: “Name” (string), “Age” (integer) , Methods “write-Name (vname: String) ”and “write-Age (vage: Integer)”. The class “TEACHER” is described by “Salary” (integer), Method “calculate-Salary (vsalary: integer)” inherits the class ”PERSON”.

Modelling respectively in UML and in CLIPS will be as follows:

In UML:



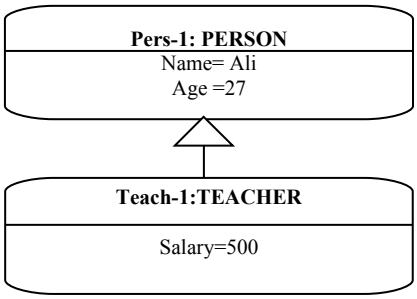
In CLIPS*:*

(defclass PERSON (is-a USER) (slot Name (type STRING) (default "Salah") (storage shared) (propagation inherit) (visibility public)) (slot Age (type INTEGER) (default 25) (create-accessor read-write) (access read-write) (storage shared) (propagation inherit) (source composite) (visibility public) (pattern-match reactive)))

(defclass TEACHER (is-a PERSON) (role concrete) (pattern-match reactive) (slot Salary (type INTEGER) (default 500) (create-accessor read) (access read-only) (storage local) (propagation no-inherit) (source exclusuive) (visibility private) (pattern-match reactive)))

- Either to create an instance of class TEACHER having the name “Ali”, age “27” and salary 500. modeling respectively in UML and CLIPS will be as follows:

In UML*:*



In CLIPS*:*

Make-instance teach-1 of ENSEIGNANT (Name "Ali") (Age 27) (Salary 500))

Conclusion:

The designer who is accustomed use UML must dominate CLIPS language, particularly the layer COOL, and how CLIPS manages its classes. It is not an easy or obvious task. In the following part, we propose our VCLIPS_UML extension.

5. Presentation of VCLIPS-UML

VCLIPS is an extension of CLIPS, it's developed in Java [1]. VCLIPS_UML offers a convivial interfacing, which on the one hand present explicitly the object concepts and on the other hand encapsulate the CLIPS language.

The principle of VCLIPS_UML is to allow an expert to describe the KB of the ES using the diagrams of classes and objects of UML and generate automatically the corresponding scripts in accordance with the language of CLIPS. The syntax of CLIPS language must be transparent for the user.

.

5.1. **Software VCLIPS_UML**

VCLIPS_UML present to the expert a main screen in Figure 3 similar to the standard usually used in the Integrated Development Environment (IDE). The main screen essentially includes two parts. The left part is the navigator of KB basis components (classes, objects, functions, rules). The right part shows the corresponding script generated automatically by CLIPS. The navigator of class's permits the selection of a component and the manipulation of its elements. VCLIPS_UML offers convivial and simple screens to define the different components of its language such as the facts in Figure 4, the templates in Figure 5, the functions in Figure 6, and the rules in Figure 7. For every step an automatic transformation in script CLIPS is displayed.

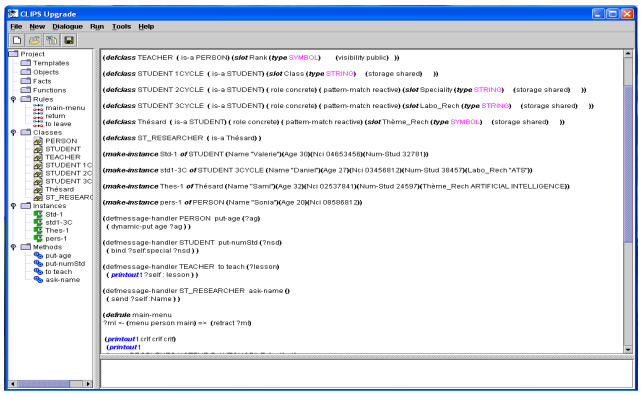


Figure 3. Principal window of the application.

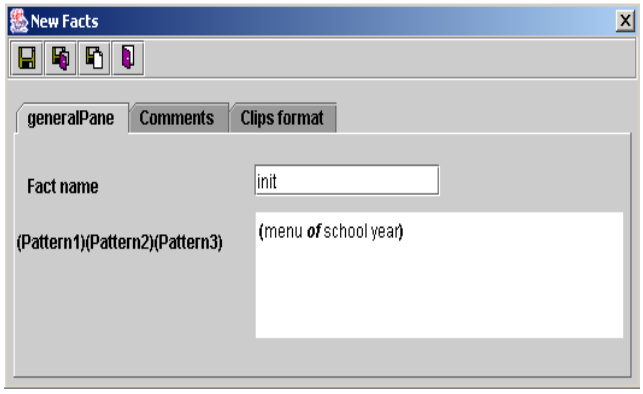


Figure 4. Data entry screen of a fact**.**

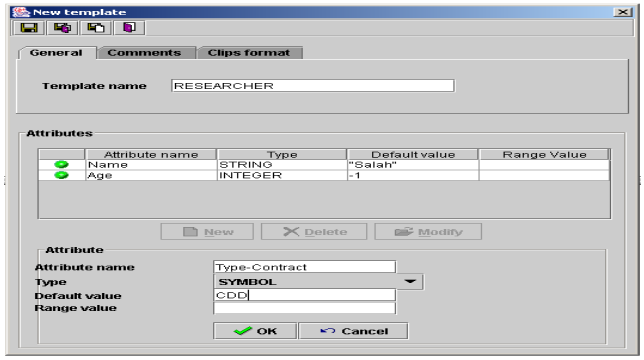


Figure 5. Data entry screen of a template.

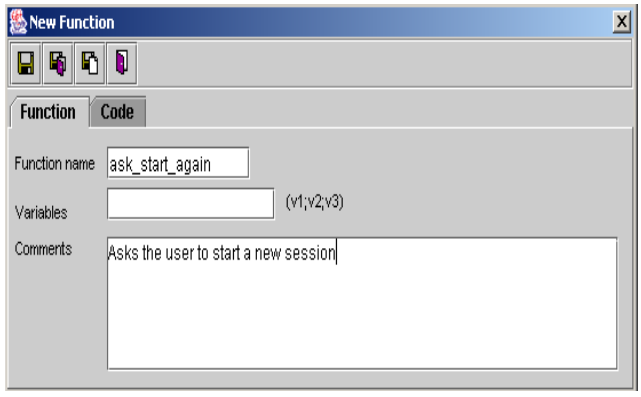


Figure 6. Data entry screen of a function.

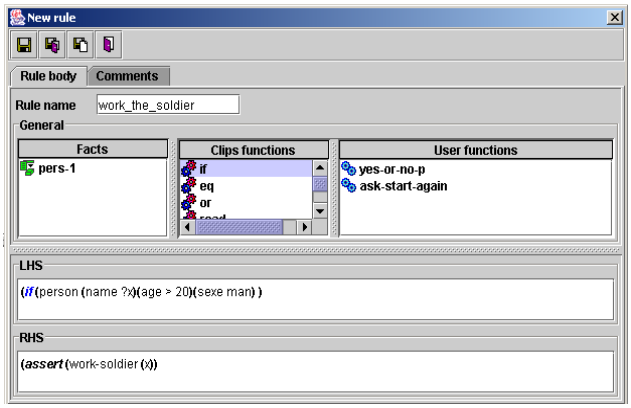


Figure 7. Data entry screen of a rule.

5.2. From UML to CLIPS

VCLIPS_UML offers to the user the modelling of his class with the UML notations in Figure 8 and its translation in CLIPS language in Figure 9. We have to note that, in Figure 8, the attributes Name, Age are in blue since they are inherited of the related class "PERSON". The attributes that are defined in addition, by CLIPS can have either the values given by default by VCLIPS_UML or manually introduced by a programmer who dominates the language CLIPS in Figure 10. To define an object of the class, the user must use the screen of the Figure 11.

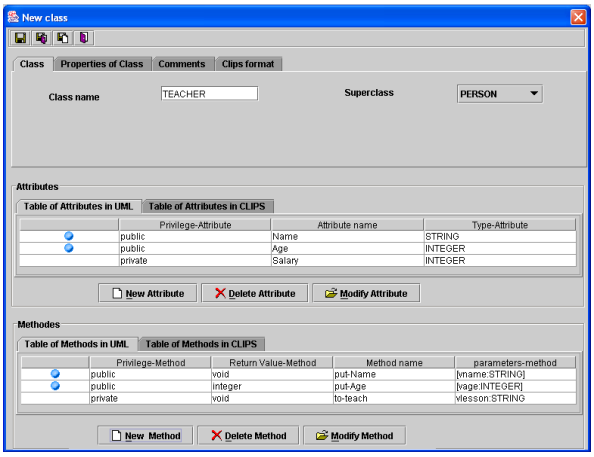


Figure 8. Data entry screen of a class (UML notation).

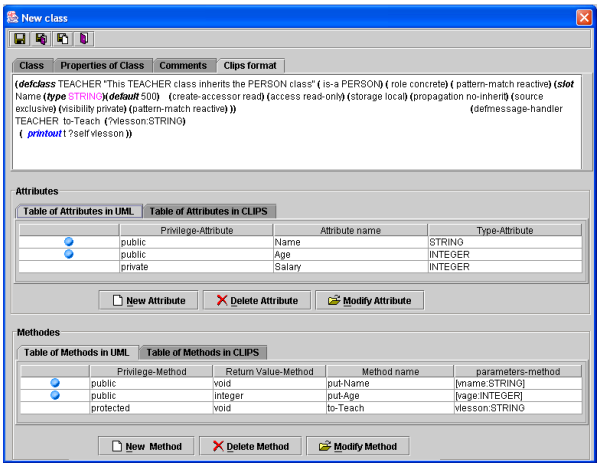


Figure 9. Data entry screen of a class (CLIPS script and UML).

Figure 10. Screen of personalization of an attribute.

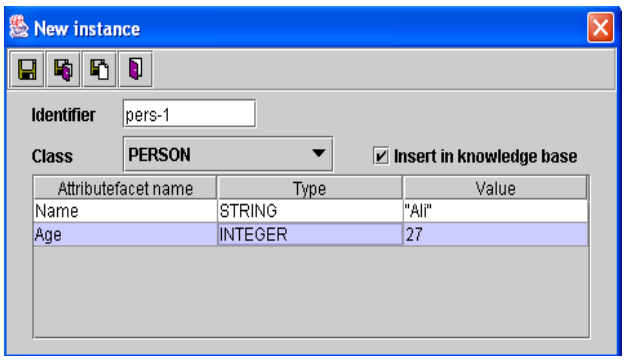


Figure 11. Data entry screen of an object.

6. Implementation of VCLIPS_UML

VCLIPS_UML is developed in Java [1]. We give in this section, the main choices done in the implementation of VCLIPS_UML.

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 4 presents an illustration of the correspondence between the CLIPS's entities and classes conceived in this application:

Table 4. Correspondence between the CLIPS's entities and the classes VCLIPS_UML under JAVA.

CLIPS's Entities	Definite CLIPS Classes for VCLIPS_UML Under JAVA
Template	Template
Fact	Template Instance
Slot	Attribute
Rule	Rule
Function	Function

To facilitate the extension and the portability of VCLIPS_UML the implementation has been developed in six packages:

- Package clips manages the interaction with the user.
- The clips.kb package contains classes that implement the KB.

- The clips.resouces package contains the requested resources files for the application under study.
- The clips.images package contains pictures used by the application.
- The clips.base package contains the utilitarian of basis of the application.
- The clips.resouces package contains the syntactic details of the CLIPS language.

It is the only package that it is necessary to modify to adapt VCLIPS_UML to another OOESS. This modification consists in replacing a file of syntax by another. This approach permits a complete extensibility of VCLIPS_UML.

7. Conclusion

In this paper, we proposed a new approach for conception and implementation of OOES based on the UML language. For this, we showed the possibility of modeling a KB using UML class and object diagrams and then we proposed an OOESS, called VCLIPS_UML, supporting UML. VCLIPS_UML is an extension of CLIPS.

VCLIPS_UML brings two principal improvements to CLIPS. The first improvement makes the concepts of the object paradigm explicit and it allows the possibility of describe the KB with the UML class and the object diagram, VCLIPS_UML generates the corresponding script automatically. The second improvement encapsulates the syntactic details of the language CLIPS which is relatively complex since it is based on the hierarchical structure of the frame. The implementation of VCLIPS_UML is made so that it can be adapted to any other EES making thus easy its extensibility and its portability on other flat shapes. A first version is operational under windows. To our knowledge such a tool was not proposed yet.

As perspectives, we essentially mention to define a tool for building an ES which makes it possible to represent all the UML diagrams when designing ES, like the rational rose tool for the OOS.

References

[1] Arnold K., Gosling J., and Holmes D., *The Java Programming Language*, Addison Wesley, New York, 2005.

[2] Bauer B. and Odell J., "UML 2.0 and Agents: How to Build Agent-Based Systems with the New UML Standard," *Computer Journal of Engineering Applications of Artificial Intelligence*, vol. 18, no. 2, pp. 141-157, 2005.

[3] Bennett S., McRobb S., and Farmer R., *Object-oriented Systems Analysis and Design Using UML*, McGraw Hill, New York, 2005.

[4] Berardi D., "Using Description Logics to Reason on UML Class Diagrams," *in Proceedings of the KI'2002Workshop on Applications of Description Logics, CEUR Electronicworkshop Proceedings*, pp. 107-118, 2002.

[5] Berardi D., Calvanese D., and Giacomo G., "Reasoning on UML Class Diagrams," *Artificial Intelligence*, vol. 168, no. 1-2, pp. 70-118, 2005.

[6] Dennis A., Haley W., and Tegarden. D., *Systems Analysis and Design with UML*, Wiley, UK, 2004.

[7] Ernest J. and Friedman E., *Jess: The Rule Engine for the Javaplate form*," http://herzberg .ca.sandia .gov/jess/, 2006.

[8] Friedman E., *Jess in Action*, Manning Publications, California, 2006.

[9] Giarratano J., *CLIPS Basic Programming Guide, Version 6.22*, http://www.ghg.net/clips / download/documentation/ Basic Programming guide.pdf, last visited June 15th 2004

[10] Giarratano J. and Gary R., *Expert Systems Principles and Practice*, PWS Publishing, UK, 1993.

[11] Grissa A., Ounalli H., and Boulila A. "VISUAL JESS: AN Expandable Visual Generator of Oriented Object Expert System," *World Enformatika Conference*, pp.290-293, 2005.

[12] Jackson P., *Introduction to Expert Systems*, McGraw-Hill, New York, 1999.

[13] Jovanovic´ J., Gasˇevic´ D., and Devedzˇic´ V., "A GUI for Jess," *Expert Systems with Applications*, vol. 26, no. 4, pp.625-637, 2004.

[14] Liao S., "Knowledge Management Technologies and Applications Literature, Review from 1995 to 2002," *Expert Systems with Applications*, vol. 25, no. 1, pp.155-164, 2003.

[15] Liao S., "Expert System Methodologies and Applications: A Decade Review from 1995 to 2004," *Expert Systems with Applications*, vol. 28, no. 4, pp. 93-103, 2005.

[16] Meyer B., *Object-oriented Software Construction*, Prentice Hall, New York, 1997.

[17] Oestereich B., *Developing Software with UML Object-Oriented Analysis and Design in Practice*, Addison-Wesley, New York, 1999.

[18] Rumbaugh J., Jacobson I., and Booch G., *The Unified Modeling Language Reference Manual*, Addison Wesley, UK, 2004.

[19] Song E., Yin S., and Ray I., *Computer Standards & Interfaces*, Elsevier Science, Holland, 2006.

[20] Tomic´ B., Jovanovic´ J., and Devedzˇic. V., "JavaDON: An Open Source Expert System Shell", *Expert Systems with Applications*, vol. 31, no. 1, pp.595–606, 2006.

[21] Willard. B., "UML for Systems Engineering," *Computer Standards & Interfaces*, vol. 29, no. 1, pp. 69-81, 2006.

Amel Touzi received the diploma of engineering in computer science and PhD in computer science from the Faculty of Sciences of Tunis, Tunisia in 1989 and 1994, respectively. Currently, she is an assistant professor at the Department of Technologies of Information and Communications in the National School of Engineering of Tunis.

Mohamed Ben Messaoud received the BSc degree in computer science from university of science of Tunis in 2003, and the Master degree in automatic and signal processing from the National school of Engineer of Tunis in 2006. His research interests include artificial intelligence, expert system, and multiagent system and also in speech processing computer-assisted learning, and computational auditory scene analysis.