A Meta-tool to Support the Development of Knowledge Engineering Methodologies and Projects

JOSÉ ELOY FLÓREZ

Computer Science Department, Carlos III University of Madrid, Avenida de la Universidad 30 Leganés, 28911 Madrid (Spain) jflorez@inf.uc3m.es

JAVIER CARBÓ

Computer Science Department, Carlos III University of Madrid, Avenida de la Universidad 30 Leganés, 28911 Madrid (Spain) jcarbo@inf.uc3m.es

FERNANDO FERNÁNDEZ

Computer Science Department, Carlos III University of Madrid, Avenida de la Universidad 30 Leganés, 28911 Madrid (España) ffernand@inf.uc3m.es

Received (03 November 2010)
Revised (03 November 2010)
Accepted (03 November 2010)

Knowledge-based systems (KBSs) or expert systems (ESs) are able to solve problems generally through the application of knowledge representing a domain and a set of inference rules. In knowledge engineering (KE), the use of KBSs in the real world, three principal disadvantages have been encountered. First, the knowledge acquisition process has a very high cost in terms of money and time. Second, processing information provided by experts is often difficult and tedious. Third, the establishment of mark times associated with each project phase is difficult due to the complexity described in the previous two points. In response to these obstacles, many methodologies have been developed, most of which including a tool to support the application of the given methodology. Nevertheless, there are advantages and disadvantages inherent in KE methodologies, as well. For instance, particular phases or components of certain methodologies seem to be better equipped than others to respond to a given problem. However, since KE tools currently available support just one methodology, the joint use of these most apt phases or components from different methodologies for the solution of a particular problem is hindered. This paper presents KEManager, a generic meta-tool that facilitates the definition and combined application of phases or components from different methodologies. Although other methodologies could be defined and combined in the KEManager, this paper focuses on the combination of two well-known KE methodologies, CommonKADS and IDEAL, together with the most commonly-applied knowledge acquisition methods. The result is an example of the ad hoc creation of a new methodology from pre-existing methodologies, allowing for the adaptation of the KE process to an organization or domain-specific characteristics. The tool was evaluated by students at Carlos III University of Madrid (Spain).

Keywords: Knowledge Engineering, Knowledge-Based Systems, Expert Systems, Software Tools, IDEAL, CommonKADS.

1. Introduction

The purpose of knowledge engineering (KE) is to design and implement computer systems known as knowledge-based systems (KBSs) or expert systems (ESs). KBSs and ESs solve problems through the representation and manipulation of knowledge in different ways, such as with frames, inference rules, or scripts. The basic structure of these kinds of systems is composed of an ontology, that is, a static representation of the world. An ontology was defined by Gruber [1] as "a specification of a conceptualization" and, generally speaking, is a representation of the elements, or concepts, of the world, together with the attributes that define it and the relationships between them. The other common element of KBSs is a set of inference knowledge, or rules, representing how to solve problems defined in this world. In KBSs, ontologies are considered to be the static (i.e., invariant) part, while inference rules represent the dynamic part. The definition of both components is achieved through a process whereby knowledge from different sources is acquired and modeled. The main source used consists of human experts with the knowledge necessary to solve domain problems in a near-optimal manner. It is the work of a knowledge engineer to extract, model, and represent knowledge in a software tool able to solve problems in a specified domain (i.e., KBSs) and, additionally, demonstrate an effectiveness similar to that of a human expert solving the same problems (i.e., ESs).

In this context, the paradox of expertise arises as one of the most important drawbacks in the implementation of KBSs. This paradox means that the more expertise a human expert has in a certain domain, the more difficult it is for the expert to explain how he solves domain problems [27]. In addition, while knowledge engineers are assumed to be rather good at modeling and representing knowledge, they do not necessarily have to be domain experts. Rather, they must only acquire relevant knowledge from domain experts. This knowledge acquisition plays a central role in KE since it is:

- Present in the whole life cycle of a KBS project.
- Very costly in terms of time and money.
- Difficult and tedious.

The integration of knowledge acquisition, conceptualization, and formalization with the classical methodological phases of any software (i.e., analysis, design, implementation, and evaluation) has led to the production of several KE methodologies, as summarized in Section 2. Attached to the majority of these methodologies is a support tool, facilitating the application of the methodology (or at least parts of it) and the corresponding development of the KBS. However, the application of a particular methodology or tool constrains the potential development of a KBS, since good elements from other methodologies and tools must be discarded [2].

This drawback of having to select a particular methodology and discard all others may be avoided with a generic meta-tool, allowing for the combination of parts of different methodologies. As demonstrated in this paper, the Knowledge Engineering

Manager (KEManager) allows the user to create ad-hoc combined methodologies for the design and development of KBSs.

Preliminary research conducted focused on how to build a generic KE meta-tool, taking into account the most common methodological phases, pre-existing tools, and a requirements specification. The conclusions of this analysis, similar to those reached in an earlier study [8], outlined how the meta-tool could cover any methodology, as well as the usability, friendliness, and effectiveness required. With these conclusions, an informal set of requirements and a high-level design of the intended KEManager architecture were proposed. Implementation of the generic meta-tool directly folowed (see Section 5 figures).

The final step in the research conducted involved the development of a use case and the administration of a user evaluation. While any methodologies could have been chosen to demonstrate the usefulness of their combination under a single support tool, CommonKADS [4] and IDEAL [5][6][7] were selected here, focusing particularly on the integration in an ad hoc methodology of IDEAL problem identification and CommonKADS conceptualization. As both IDEAL and CommonKADS methodologies are taught by the study authors in their courses at Carlos III University of Madrid (UC3M), their selection permitted an evaluation of KEManager by select UC3M students.

Since the first prototype became available in 2008, KEManager has been used by students at UC3M and the Universidad Nacional de Educación a Distancia (UNED). At the former institution, KEManager has been used as part of a fourth-year undergraduate course in the Computer Science and Engineering degree program. A set of polls and questionnaires were distributed to the students in order to obtain feedback about errors, potential improvements, general user satisfaction, as well as other considerations related to the KEManager. This data set indicates the degree of effectiveness achieved by the tool and points the way to future improvements and modifications to be made.

The remainder of this paper is divided as follows. In Section 2, related work is discussed. Section 3 offers traditional definitions for common methodological phases, as well as a brief look at currently-existing KE tools. Section 4 informally lists software requirements and indicates the high-level architecture of the KEManager. Section 5 explains the implementation and principal features of the meta-tool. Section 6 presents KEManager results in the use case with IDEAL and CommonKADS, obtained from evaluations conducted by UC3M students during the course of two academic years. Finally, Section 7 draws conclusions from study and evaluation findings and discusses areas for future research.

2. Related Work

A methodology is defined as a set of steps or phases for the achievement of an objective. In other words, "a methodology describes how a project is organized, the order in which the tasks are executed and the interfaces defined between them" [9].

In the context of KE, a methodology can be defined as a set of techniques, procedures, and tools to acquire, represent, implement, validate, integrate, and maintain the knowledge inside in an ES. There are several methodologies defined to work in KE that come from traditional software engineering methodologies [3]. Originally, methodologies were based on prototyping, an encode and repair approach. Later, methodologies had cascade life cycles in which the input of each phase is the output of the previous phase. With such a methodology, when errors are detected, it is possible for them to be fixed in previous phases without starting from scratch. Nevertheless, all requirements must be defined at the beginning of the project, something that is not possible with KE. Another alternative used is the spiral life cycle. In it, all phases are repeated to generate intermediate products that are evaluated to define new requirements, evaluate costs, and select the best alternative for the following cycle. In IDEAL [5][6][7], an extension of this approach known as conical, or 3D spiral, life cycle is represented, allowing the definition of new requirements when the system is deployed. Table 1 presents the chronological development of these methodologies.

Table 1. Methodologies and Tools.

Name	Year	Phases	Class	Tools Related
Harmon et al. [10]	1985	6	Prototyping	-
Waterman [11]	1986	5	Prototyping	-
ICOT [12]	1986	4	Prototyping	ICOT [12]
Carrico [13]	1989	8	Prototyping	-
Alberico and Mico [14]	1990	2	Waterfall	-
POLITE [15]	1991	6	Waterfall	-
KADS [16]	1992	6	Waterfall	PCPACK [30]
I.D.E.A.L. [5][6][7]	1995	5	Conical	REPGRID[29], PROTEGE [31]
MOKA [17]	1998	6	Spiral	PCPACK [30]
CommonKADS [4]	1999	6	Spiral	PCPACK [30], PROTEGE [31]

One of the first implementations of a tool for KE was KEDE [18], developed with Common Lisp using different knowledge representation formalisms (KRFs) with the aim of increasing the number of solvable problems for a KBS produced with the tool. In a study by Huneault [19], different KE tools are described and classification criteria, from the KRF used (mainly rule-based and object-oriented) to the estimated building cost of a KBS in certain environments (mainly power plants), are introduced. The high cost of building KBSs and the losses in quality and productivity observed, mainly due to the subjective selection of tools and rep-

resentations, brought attention to the need for a structured method for the building and integration of knowledge modeling in KBSs.

A study performed on nuclear power plants [20] shows the importance of knowledge acquisition and the verification and validation of the resulting knowledge base to produce reliable KBSs through the implementation of a knowledge acquisition tool. Additionally, ontologies have been gaining importance [21] in the representation of knowledge bases inside KBSs.

Focusing on the graphical user interface (GUI), the acquisition tool, Protégé [22], presents the concept of a component or plug-in to extend and adapt the tool such that it may be more effective with a greater diversity of problems and domains. The knowledge acquisition process has also been applied in other AI disciplines such as automated planning [23] through a graphical tool designed for the acquisition, validation, and maintenance of planning domain models.

Tagger [24] is a real-time tool allowing the user to define tags at different levels of abstraction during a knowledge acquisition session. The authors give a more specific description of the knowledge acquisition bottleneck mentioned in the introduction and stress the importance of the knowledge acquisition session in itself, in addition to the further processing.

In Chan [25], the design and implementation of a software tool based on the inferential modeling acquisition technique is demonstrated for the construction of ontologies in an application domain. The study emphasizes the importance of acquired knowledge reusability in order to solve different problems in the same domain, thereby alleviating part of the knowledge acquisition bottleneck. XML is also proposed as a way to import and export information to and from the tool.

The Knowledge Engineering Suite [26] is a tool for working with ontologies in the Semantic Web. One of the Suite's principal features of interest is that it permits several knowledge engineers and experts to work at the same time.

All tools reviewed above cover particular phases of the KE process or are designed to work in given domains. It is the objective of the present study to allow for the integration of the advantages offered by these tools through the design of a meta-tool capable of suporting several KE methodologies and adaptable to different domains.

Considering the problems that the previously-cited authors had set out to resolve and the choices made to develop the corresponding tools to work in KE, this study is justified for the for basic improvements it proposes for KE: (1) a more effective knowledge acquisition process, (2) a more proper way of representing, managing, and reusing knowledge, (3) a mechanism to improve, adapt, and extend previously-discussed issues, and (4) a more effective interaction between the system and the user (i.e., GUI). The overarching goal of the study is the development of a meta-tool (i.e., generic framework) that works in KE and in which the user may combine methodologies, acquisition techniques, and representation formalisms together with a proper way to interact with these through a well-defined and accessible GUI.

3. Motivation

The present section attempts to determine the essential features that must be implemented by a software tool such that the majority of KE methodologies are addressed. In addition, we describe some of the available tools.

3.1. *Methodological Phases*

In order to effectively deal with any methodology, a software tool must implement problem identification, knowledge acquisition, and conceptualization phases. It is the goal of this sub-section to define how these phases must be implemented in a generic meta-tool.

In problem identification, the problem to be solved is analyzed. Implicit in this analysis is the identification of the domain in which the problem will be solved, the users, the data to be managed, the resources available to perform the knowledge acquisition process, and the results expected. Problem identification is often divided into several goals, each of which having a corresponding method for assigning quantitative values to a set of questions. Among these sets, questions are often included regarding the KBS requirements, the nature of the problem (e.g., Is the problem domain a good candidate to be addressed using KE?), the definition of the problem features, and, at a high level, how to achieve the solution.

Once the responses to these questions are set, a numerical value is generated, determining if the problem may be solved using KE. Several numerical evaluation methods have been defined that can be applied to question sets.

Given the characteristics defined above, the problem identification phase requires that the user be able to add, modify, or delete the questions to be grouped into goals, as well as to select the method whereby the responses to these questions are aggregated. Therefore, it is also necessary that the tool provide a mechanism whereby the question set may be defined and the outgoing final numerical value be obtained according to the methodology selected.

In a set, every question possesses a numerical weight influencing the final result. Depending on the methodology applied, the minimal value to be reached in the evaluation process such that KE might be considered a valid solution to a particular problem will be determined by different weights, data types (i.e., binary, ordinal, nominal, or numerical), and thresholds. Finally, some questions are compulsory, while others may be optional.

The second methodological phase of the tool addresses the question of how to define the knowledge acquisition techniques to be applied. Such techniques are similar for all methodologies, implying the acquisition and processing of data from different sources such as written texts and human experts. As explained in Section 5, this study considers some of the most commonly-used acquisition techniques including knowledge extraction from texts, open and structured interviews, triadic method, and the repertory grid.

Knowledge acquired in this phase must be stored. While this knowledge is most

often stored as a text, different structures are often used for each technique. For example, an open interview can be seen as a tree of questions and answers. As a result, the tool must include a functionality to search for and represent concepts, terminology, attributes, and relationships in the text, as well as use the text to model both the static and dynamic aspects of this knowledge. Furthermore, since the knowledge acquisition activity is a modeling activity, the tool requires a subsystem to specify models graphically. Section 5 of the present study describes the implementation of each acquisition technique in KEManager.

Additionally, the decomposition of the entire problem and solution conception is, for all intents and purposes, a preliminary modeling process. Therefore, it is necessary to include in the software tool a graphic modeling sub-system together with a set of components to specify the required textual information.

The third methodological phase included in KEManager is that of knowledge conceptualization. The output of this phase is a formal specification of the final system avoiding implementation details. The principal aspects to be defined in this phase are the following:

- The statical representation of the model, or, in other words, concepts, attributes, and relationships defined between concepts composing the statical definition of the knowledge, together with the instances of such elements, which represents the ontology.
- The interaction between the system and the real world.
- The definition of the reasoning process using the abovementioned knowledge to solve particular problems.

As the conceptualization phase is, essentially, a modeling activity, the software must work with different types of representations such as CML2 (CommonKADS suggestion) and UML.

3.2. *Available Tools*

Having briefly analyzed the different phases to be implemented by the meta-tool, this sub-section presents and discusses the most important software tools available for working with KE, focusing on the positive and negative aspects of each. While some tools work with only certain aspects of KE, such as RepGrid [29] for making grids and semantic networks, or ModelDraw for making CommonKADS modeling diagrams, other tools like PCPACK [30] and Protégé [31] implement almost the entire process. For this reason, the analysis of the following paragraphs is focused on these latter two tools.

PCPACK is a tool designed for the acquisition, use, and sharing of knowledge inside a corporation. A project can use a set of predefined types of ontologies including: "empty" with only concepts, attributes, and relationships; "standard" which, in addition, includes tasks, task components, and values; "MOKA" [17] based on MOKA KE methodology; and the "General Technical Ontology" (GTO). Knowl-

edge acquisition is covered by the tool through a sub-system allowing the user to mark pieces of a text by selecting different colors, each being associated with an ontology element (i.e., concept, attribute, relationship, or value). The modeling process is performed using different types of diagrams and representations, and is conceived of as a sequence of knowledge acquisition and modeling steps. At every step, the ontology can be represented in various formats to make it accessible for the rest of the organization. Figure 1 shows the knowledge acquisition form, "Protocol tool", allowing the user to analyze texts identifying words or sentences as categorical attributes, concepts, or tasks. In Figure 2, the reasoning process is displayed in a diagram connecting tasks with concepts, and indicating the temporal development of the process.

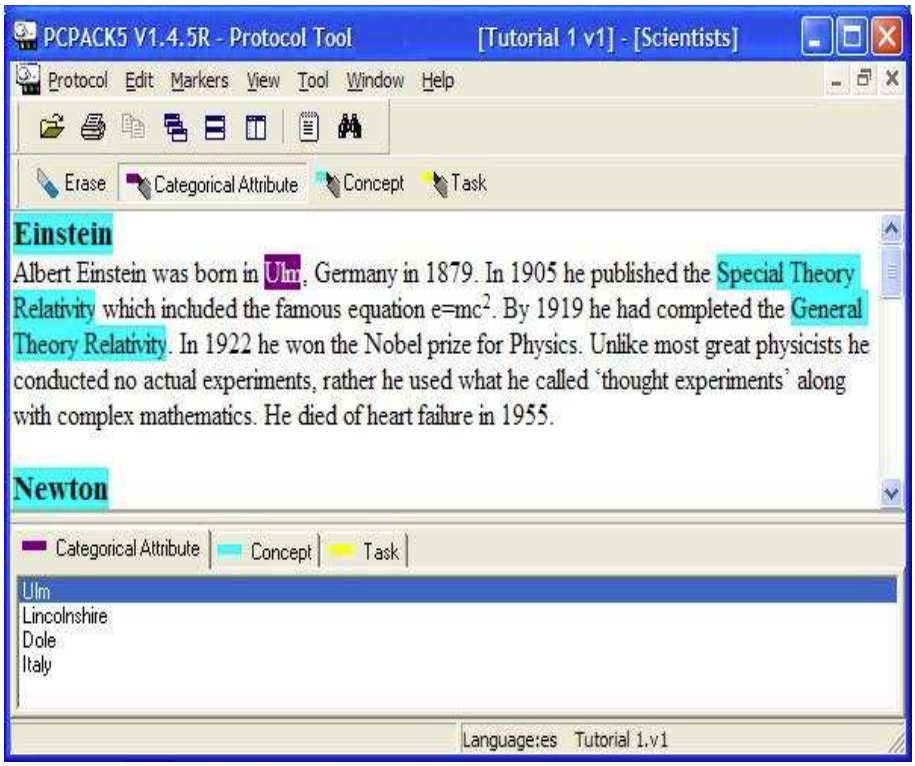

Fig. 1. Text analysis tool.

The testing of the tool reveals several interesting and useful features from the perspective of the design proposed in this study. One example of this is the representation of knowledge in XML, a GUI allowing the user to model almost all knowledge graphically. The following points indicate what testing reveals to be the three most important features to add to the proposal design:

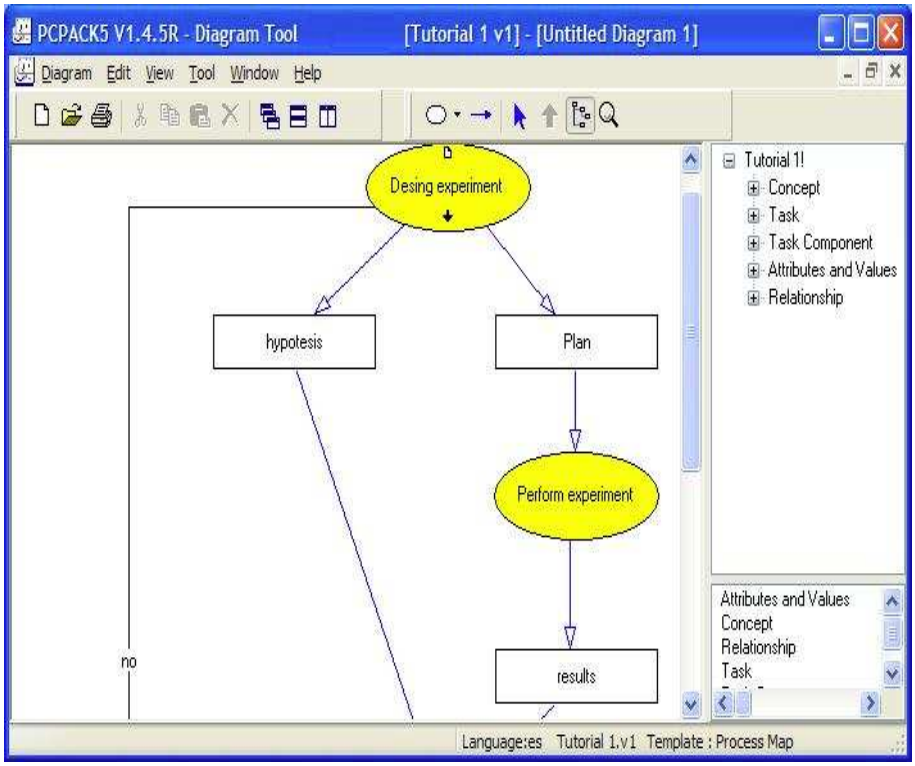

Fig. 2. Reasoning process diagram.

- A text analysis tool to extract concepts, attributes and relationships.
- The use of XML, considered particularly apt for knowledge and project representation.
- A graphic modeling tool that is powerful and easy to use.

It is also important to note that the GUI was found to be simple and friendly, offering many "good manners" for the design of the proposed application.

Protégé [31] is a free, open-source tool developed at Stanford University for the acquisition and representation of knowledge using ontologies. While originally conceived to work in medical domains, Protégé has evolved to work with general purpose ontologies and includes more advanced features for knowledge manipulation and visualization. Protégé permits the user to specify the task which can be reusable in the future. Just as with PCPACK, Protégé supports work with knowledge in various formats (e.g., XML). Moreover, it offers a very useful GUI for the addition, manipulation, and visualization of knowledge through graphical components. Perhaps the most important feature to be taken into account here is the possibility to extend tool functionalities using external plug-ins. Figure 3 shows the Protégé GUI.

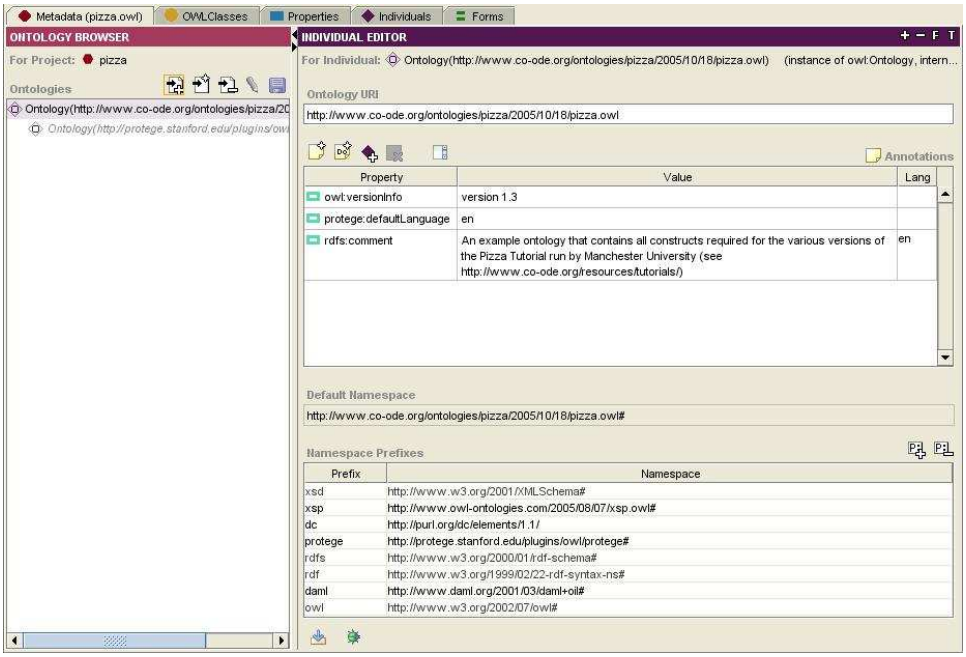

Fig. 3. Protégé GUI.

Following the review of the tool, the following characteristics of Protégé were identified for inclusion in the KEManager:

- The tool is expandable and modifiable through plug-ins, using an API to add new types of ontologies, knowledge base specifications and acquisition techniques.
- Existing ontologies can be added.
- Knowledge representation is based on standards such OWL and XML Schema.

Following the analysis, some general conclusions may be drawn about the state of the art. Firstly, not many software tools are actually available to work in KE (with the majority having been developed by university research groups). Secondly, of the tools available, most focus on knowledge representation and ontology design, while ignoring knowledge acquisition and problem identification, two phases requiring costly interaction between the expert and knowledge engineer. For instance, the tools studied do not support the implementation of knowledge acquisition techniques, such as repertory grid, with an appropriate GUI in order to reduce time and cost. Taking into account these conclusions and other information presented in this section, a set of requirements in a high level design were informally specified to serve as input in the KEManager implementation process. Both these requirements

and the KEManager design are described in the following section.

4. Requirements and Design

To begin, it is important to first acknowledge particular implementation considerations made prior to the specification of requirements. One important consideration regarded the formalism selected for knowledge representation. As explained above, it was determined that XML is the best choice for the storage and management of data due to its widely accepted use and the existence of numerous tools for XML in several programming languages. In addition, XML separates information from the graphical representation or interpretation the user desires to select.

As mentioned earlier, the great size of the undertaking of implementing a comprehensive tool to work with KE leads to the identification of three priorities for the tool: that it be configurable, adaptable, and scalable. In this way, any functionality could be extended in the future. Consequently, the tool was conceived as being open and in which knowledge acquisition techniques and KE methodologies can be added easily.

The requirements specified for the tool were grouped into five categories and specified in a manner similar to that used in software engineering. The principal requirements from these categories are summarized below:

- The system must be a powerful and easy-to-use GUI in which the majority of the work can be performed using graphical components.
- The system must properly implement the knowledge acquisition process through a set of knowledge acquisition techniques, as described in Section 2.
- The system must properly handle methodologies, understanding a methodology to be a set of interrelated steps or phases performed in a specific order.
- The system must support graphic modeling, including tools for the creation of diagrams and the capability to attach semantic meaning to the components included in these diagrams.
- The system must be extensible in three ways; namely, it must support the addition of (1) new knowledge acquisition techniques, (2) new methodologies and phases, and (3) new types of graphical representations (i.e., diagrams).
- The information must be represented in a well-known format such that other software tools may access this information to perform additional tasks.
- The tool should be able to generate printable documents from the knowledge acquired in various formats (e.g., PDF or RTF).

According to these requirements, Figure 4 presents the model defined to work inside the KEManager. The principal concept is the project, modeled as a set of

knowledge acquisition sessions together with a set of phases associated with a methodology, in which information is acquired and managed principally through graphical modeling. Figure 5 shows the package decomposition of the meta-tool. The main control implements the framework through modules, which can be extended to add knowledge acquisition techniques and methodologies, together with tools capable of providing diagrams and analysis of texts. All these extensions can be added using the functionality implemented in the extension module. The Web deployment represents a way to make the application accessible through a Web browser.

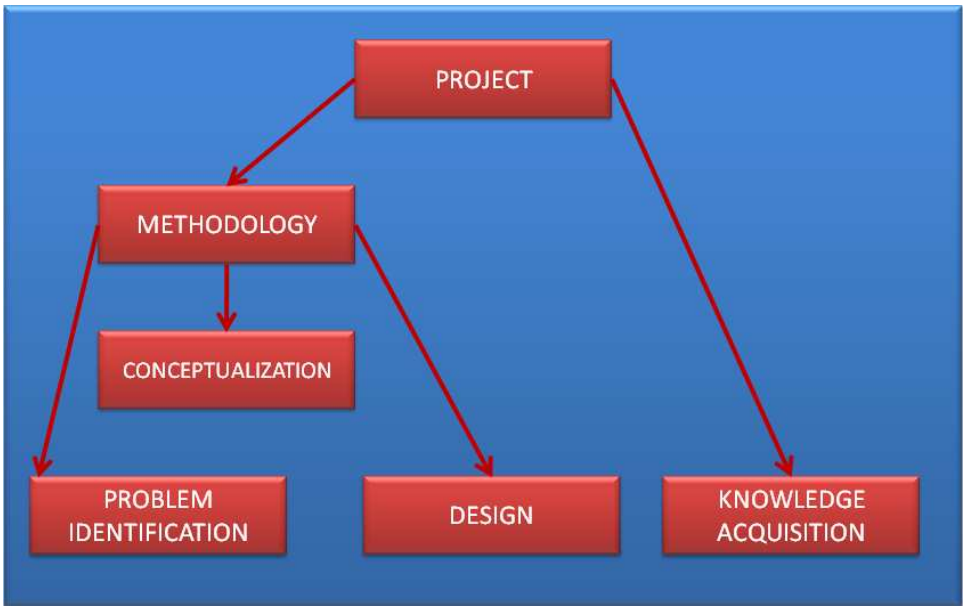

Fig. 4. Model inside the meta-tool.

The manner by which the functionality of KEManager is extended is based on Java's dynamic class loading capability (generally present in almost every interpreted language), allowing the loading of new code during the execution of a program. Using XML to define how the new components must be loaded, programmers must derive a set of specific classes, depending on the component which they want to add, from the generic classes defined in the framework. GUI representation and behavior for the specific component must then be specified. Figure 6 illustrates the steps of this process.

In the first column of Figure 6, the elements that can be defined using the framework are shown. The second column presents the elements (i.e., classes) defined in the core of the framework for the inclusion of new acquisition techniques, methodologies, and phases, or diagram figures. For the acquisition techniques, three classes

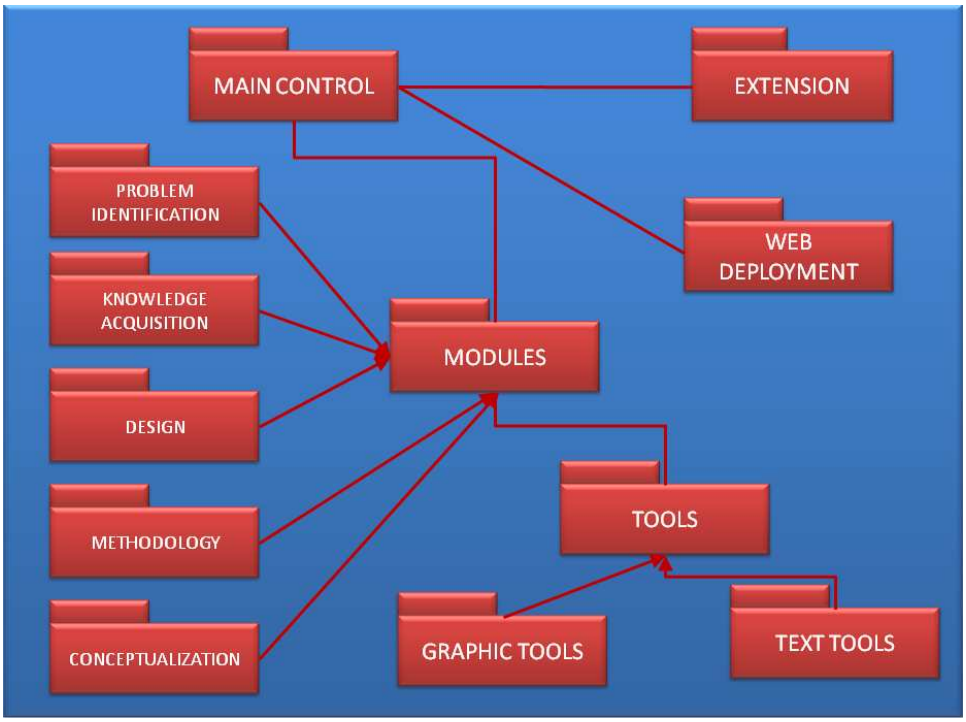

Fig. 5. Package decomposition.

are defined: one for the GUI (AcquisitionFrame.java), one for the data managed (AcquisitionSession.java), and, finally, one for the representation (AcquisitionRepresentation.java) in printable formats. Three basic classes have been defined for the phases, having the same purpose as the abovementioned classes. For graphical components, a class (BasicFigure.java) exists implementing the behavior common to each.

The third column of the table presents derived classes for the implementation of concrete elements. For instance, to create the grid knowledge acquisition technique, three classes were derived from the classes defined in the core (GridFrame.java, GridSession.java and GridRepresentation.java). Finally, a rectangle (Rectangle.java) was derived from the class defined in the core (BasicFigure.java) to create the corresponding graphical component.

The final column shows the XML specification for the loading of each extension type. This specification is used by the extension module to load each element into the framework; for example, to add a new knowledge acquisition technique, it is necessary to create a new XML tag to define the name (Grid), the frame implementing the GUI (GridFrame), the class to manage the data (GridSession), and the class for printable representations (GridRepresentation).

The XML structure for the definition of methodologies is very similar. Firstly,

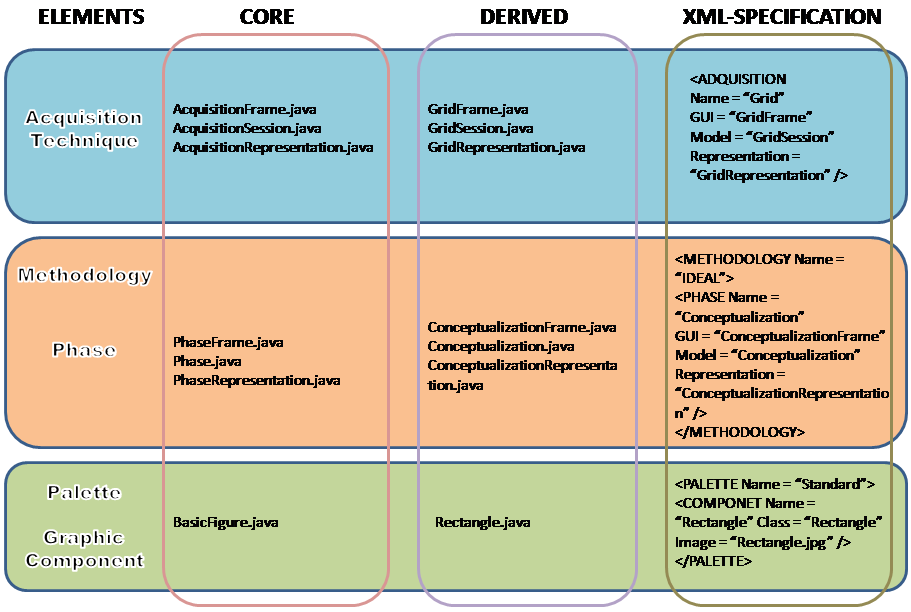

Fig. 6. Component specification process.

an XML tag must be defined to specify the name of the methodology. In this tag, XML tags for each phase are defined in the same way as knowledge acquisition sessions. For example, there is a conceptualization phase which has defined the GUI (ConceptualizationFrame), the class to manage data (Conceptualization), and a class for representation (ConceptualizationRepresentation). This phase specification is included in a methodology, "IDEAL". The structure allows the user to create methodologies using previously-created phases.

Finally, the XML structure to define graphic components includes an XML tag in which the user must define the name (Rectangle), the class which implements the component (Rectangle), and the name of the image to be shown on the GUI diagrams tool. Graphical components are grouped into palettes with an XML tag to specifying their names. Components can be shared across palettes in the same way that phases can be shared across methodologies.

5. Implementation

In this section, the two-step development process for the meta-tool is described. In the first step of the process, the framework and common elements of the system were implemented. Included among these common elements were the main window

of the GUI, the external tools providing specific functionalities, and all base classes required for the definition of plug-ins with new methodologies, phases, knowledge acquisition techniques, and graphic diagram elements.

Figure 7 shows the main window of the system. The main element defined inside the meta-tool is the KE project. A project is a representation of all information gathered with all tasks done, all diagrams defined, and other additional information, depending on the particular KE methodology.

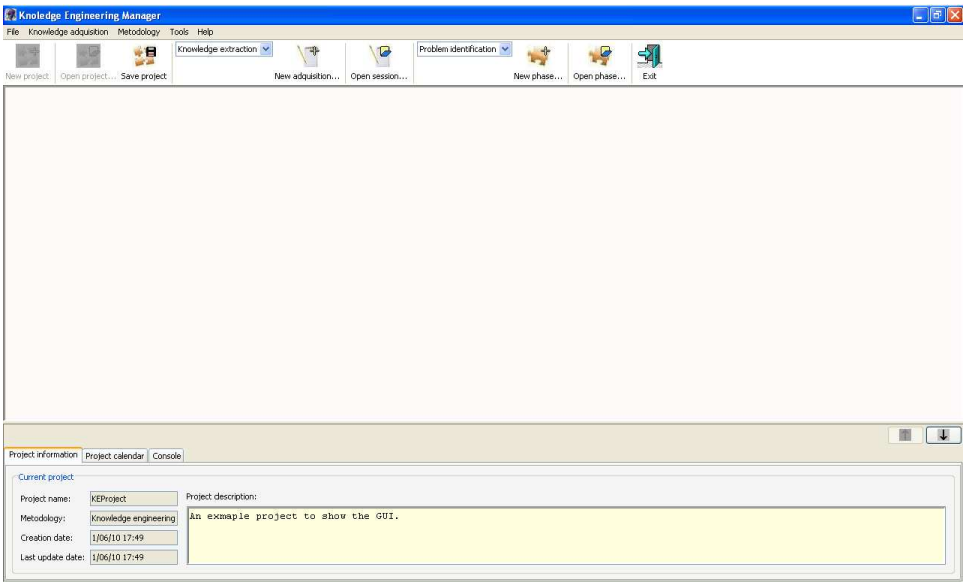

Fig. 7. Main window of KEManager.

When the user creates a new project, it is necessary to specify the KE methodology to be used. This selection determines the phases with which the user will work. Whatever the choice, however, the user may access all knowledge acquisition techniques implemented. As explained in the introduction, the knowledge acquisition phase is shared by all methodologies and present (with a particular degree of importance) throughout the entire life cycle of the project. Figures 8 and 9 show the form implemented to work with a KE phase and knowledge acquisition technique. In both examples, the information presented in the GUI is grouped into tabs. Each tab is related to a specific step in the knowledge acquisition technique or phase.

Every phase or knowledge acquisition technique may have graphic diagrams. For instance, Figure 10 presents an example of a diagram with an evaluation task specified in the CommonKADS methodology.

The second step in the meta-tool implementation involved the specification of the KE methodologies, knowledge acquisition techniques, and diagrams palette. As

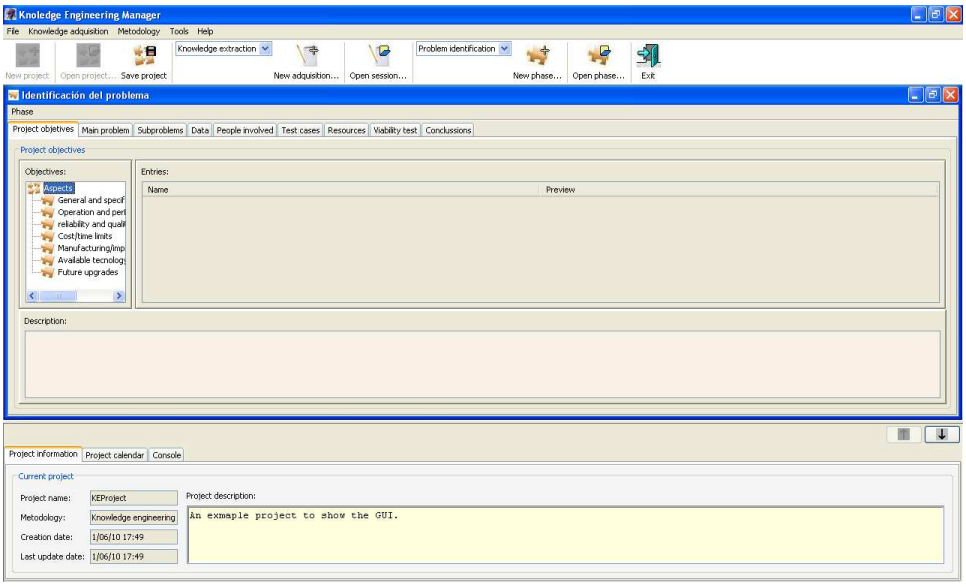

Fig. 8. An example of a KE phase.

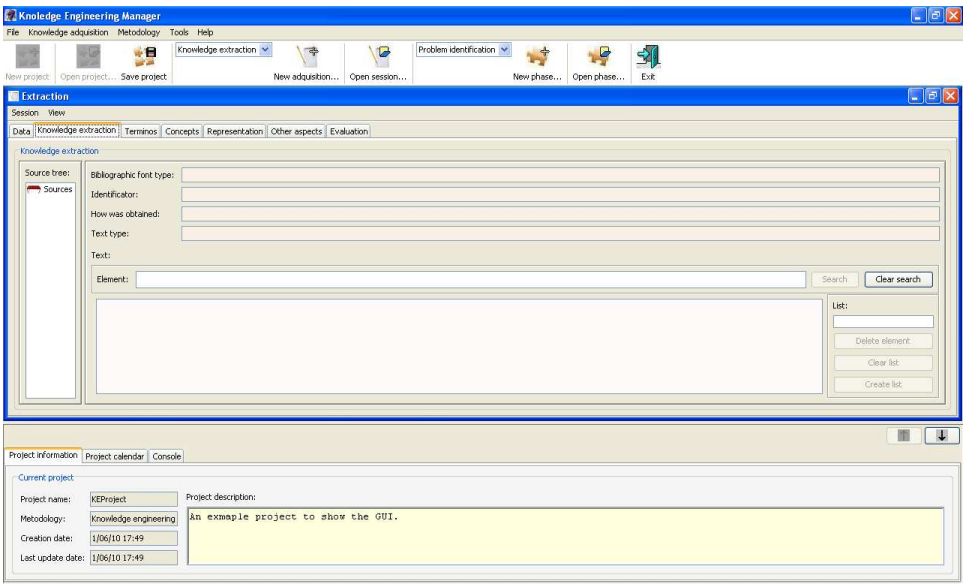

Fig. 9. An example of a knowledge acquisition technique.

explained in Section 2, the KE methodologies currently specified in the meta-tool are IDEAL and CommonKADS. The diagrams defined were those required to work

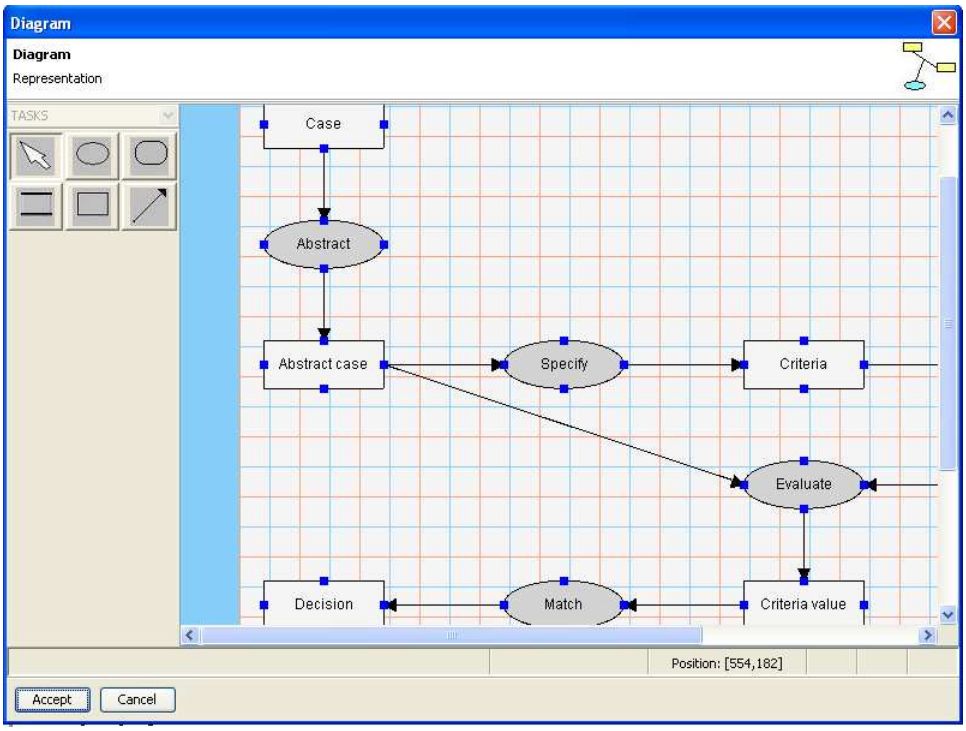

Fig. 10. An example of a diagram.

with both KE methodologies together (with principal components in UML). The knowledge acquisition techniques implemented were the following:

- Knowledge extraction from texts (see Figure 11).
- Open interview (see Figure 12).
- Structured interview (see Figure 13).
- Triadic method (see Figure 14).
- Grid (see Figure 15).

All knowledge acquisition techniques share the same tabs, save the one implementing the acquisition session. In knowledge extraction, the GUI allows the user to specify different text sources and analyze the texts used. The open interview session was modeled as a tree of general questions in which an initial set of questions was defined. Other questions derived from this set may be considered following the response given by the expert. In the structured interview, the session was modeled as a set of short questions grouped according to the objective pursued by each. The triadic method shows a list of objects and a table of three elements selected, together with an explanation of why two of these elements were grouped together and the third kept separate. Finally, the grid illustrates a way to create a grid session

by specifying the objects and attributes, and setting different values for each pair.

All other tabs are common to the knowledge acquisition techniques implemented and allow the user to acquire, model, and represent the knowledge specified during the session; for example, defining new concepts and terms, creating diagrams, and specifying rules.

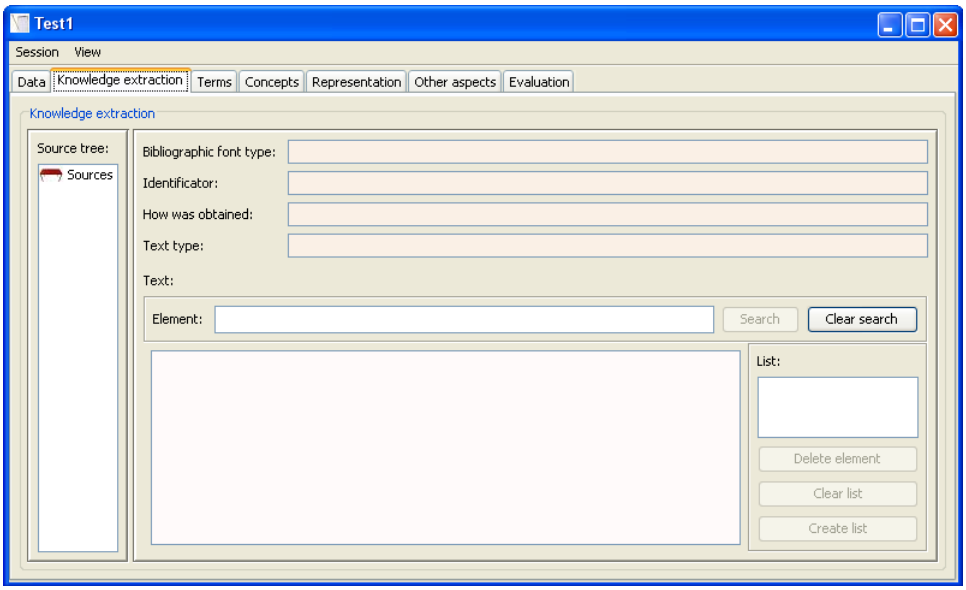

Fig. 11. Knowledge extraction.

In order to cover the conceptualization phase specification in CML2, the CML2Editor, a previously-released tool (and CS senior project of David García, supervised by study author Javier Carbó) for the editing and syntactical analysis of files in CML2 was included in the meta-tool. Following the completion of the framework implementation, the following functionalities were achieved:

- A fully operative tool allowing the definition of KE methodologies, acquisition techniques and diagram representations through plug-ins.
- A complete easy-to-use GUI for interaction with the user.
- An easy way to manage KE projects, regardless of the methodology selected.
- Multilingual support.

The meta-tool described here is publically available on SourceForge[32]. It can be downloaded, evaluated, and used without restrictions. Once made public, the meta-tool was distributed among KE students at UC3M in order to test its effectiveness and receive feedback. This feedback allowed the authors to further evaluate and

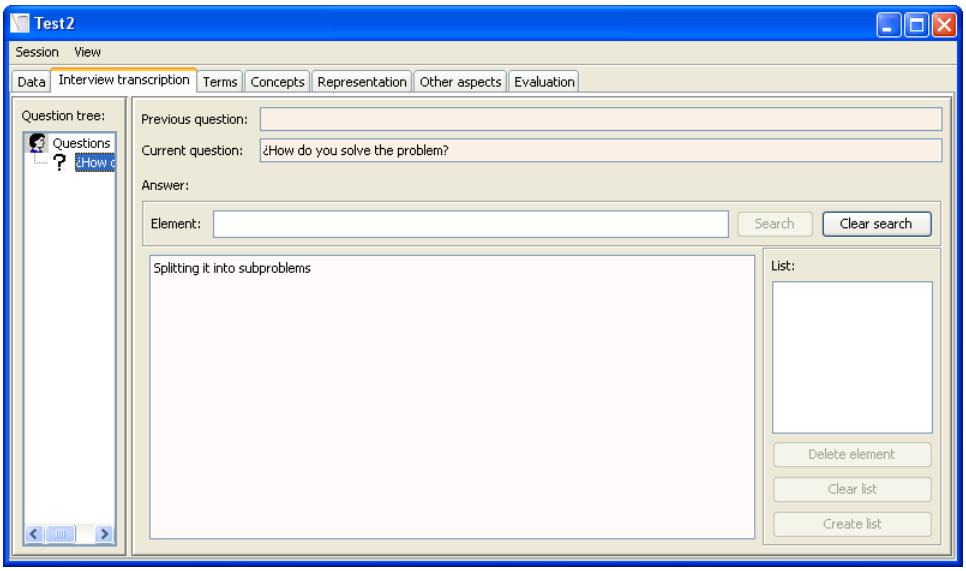

Fig. 12. Open interview.

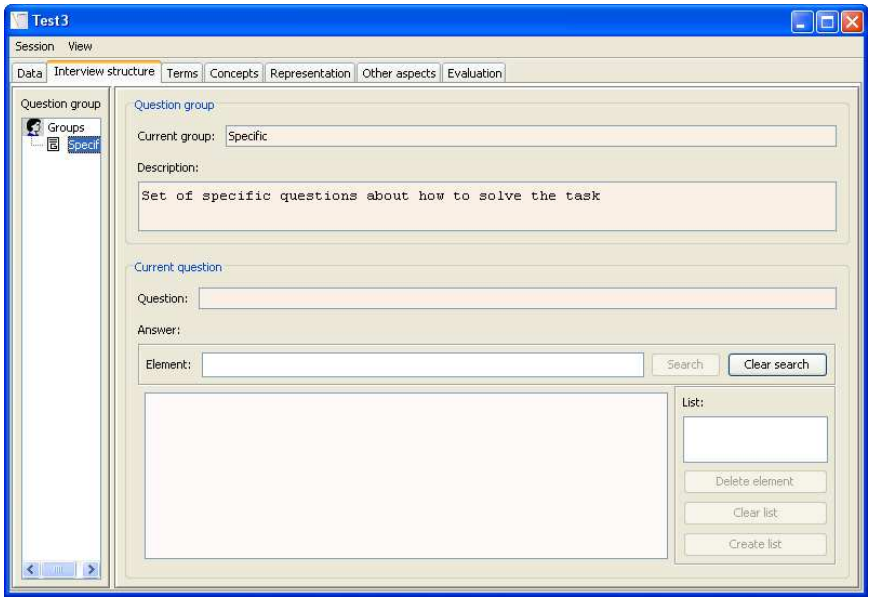

Fig. 13. Structured interview.

improve the meta-tool. The following section presents and discusses the evaluation results obtained.

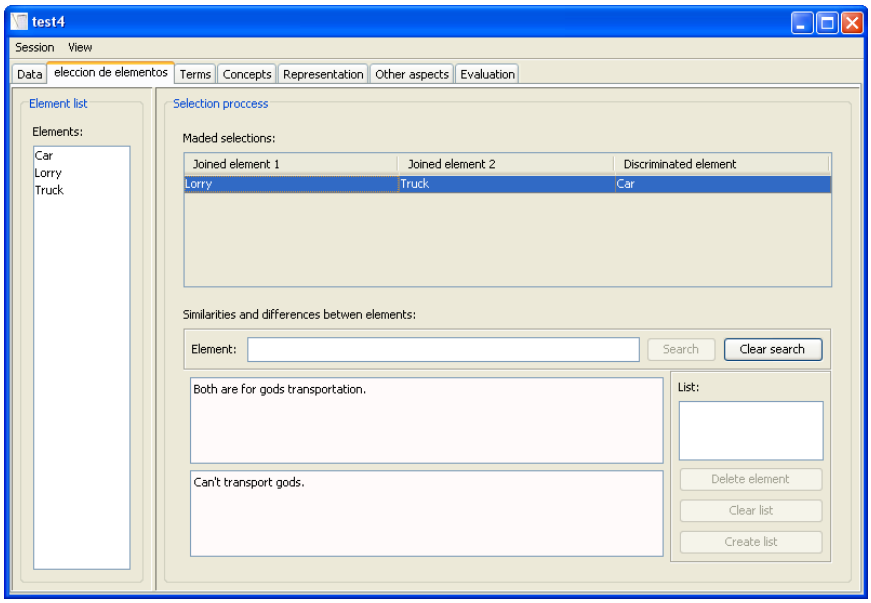

Fig. 14. Triadic method.

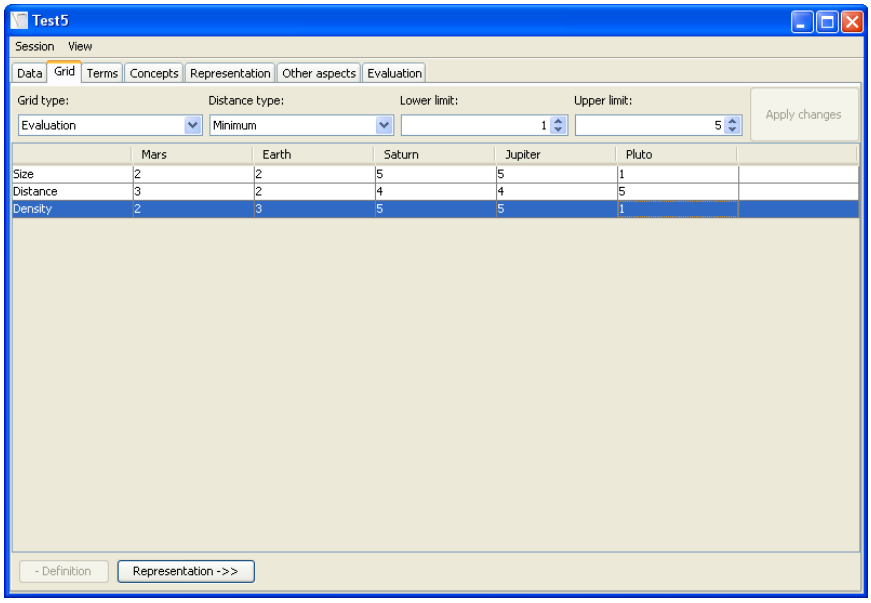

Fig. 15. Grid.

6. Use Case and User Evaluation

In this section, the development and implementation of a new methodology, a mix of IDEAL and CommonKADS methodologies, with KEManager is analyzed. In the

creation of this new methodology, the two pre-existing methodologies were selected due to the evaluation procedure presented here; namely, using UC3M undergraduate KE course students, dozens of whom were already familiar with KEManager which they had used for the development of course-long KE projects. The two methodologies were integrated, taking the problem identification evaluation method and knowledge acquisition techniques suggested by IDEAL, and the knowledge and communication models produced by the conceptualization phase as suggested by CommonKADS. The following sub-sections describe in greater detail this integration and the corresponding evaluation performed.

6.1. *Use Case: IDEAL and CommonKADS*

In the IDEAL methodology, questions for problem identification are grouped according to the following factors:

- Plausibility: Is the implementation of a KBS possible in this context?
- Justification: Assuming that the implementation is possible, is it the best option?
- Adequacy: Is a KBS adequate in order to solve the problem?
- Success: Is the context favorable for the success of the KBS?

In CommonKADS, conceptualization involves graphical and formal representations of task, inference, and conceptual models. KEManager, through the use of diagrams, allows for the graphical representation of ontologies, annotated inference structures, and task decompositions. These diagrams can be formalized in concepts, relationships, structures, roles, and tasks using UML or CML2. To support this formalization, KEManager includes a text editor (CML2Editor) to syntactically verify CML2 code with more user-friendly and context-dependent assistance for syntax errors than that offered by the KADS22 official compiler.

It is important to emphasize here that the principal benefits gained in the use case by this particular integration of IDEAL and CommonKADS methodologies are directly due to the limitations, time restrictions, and general comprehension of methodological phases of the use case student evaluators. In IDEAL, problem identification is fast, easy-to-understand, well-structured, and particularly apt for use given the KE course materials and semester-length time frame. Similarly, as CommonKADS is one of the most widely-used and well-known KE methodologies, focusing on the knowledge model prevents students from getting overwhelmed with the other models (e.g., agents, organizations, tasks, and design). While the particular integration used here may have been the most expedient given the practical considerations of the use case and student evaluators, the authors do not suggest that the integration provides for a better KBS than a different combination of methodologies. In fact, it is important to remember that the principal goal of this study is to present a meta-tool capable of combining phases from any of a number of different methodologies. The combination explained here should be taken, then,

merely as an example of this possibility as provided by KEManager.

In the KE course taught at UC3M, individual KE projects are developed by two-student groups. Particular KE domains dealt with have included city power requirement predictions, diagnoses of dermatological conditions of the face, organization of dance performances, basketball coaching, and investment consultancy. Each student group is responsible for locating and consulting with a real-world expert for its particular KE domain.

Although it would have certainly been informative to have compared the quality of the KBSs produced by student groups with and without KEManager, such a possibility was ruled out for this use case due to specific characteristics of the course and of KE projects, reflecting particular pedagogical aims. Firstly, in order to expose the students to the different sorts of problems that can be addressed by KE (e.g., classification, diagnosis, scheduling, and monitoring), different KE projects were undertaken by each student group. Secondly, to give students the experience of having to interview real-world experts to acquire the knowledge necessary for their projects, different experts were consulted by each different group.

As the student projects evolved during the length of the course and each project involved most of the main aspects of the complete process, students made extensive use of the tool throughout the semester. Consequently, the students gained a familiarity with the tool, allowing them to provide qualified opinions about it and suggest ways in which it could be improved.

6.2. *User Evaluation*

The KEManager meta-tool has been used by UC3M and UNED students since 2008. Two different meta-tool evaluation questionnaires were administered to KE students at the end of the semester-long course in 2009 and 2010. While the use of KEManager was never compulsory for the students, questionnaire responses indicated that KEManager was used by the great majority, most of whom indicated its usefulness in the acquisition process. In the first questionnaire, administered to students in 2009, questions focused exclusively on technical and operative aspects of the tool in order to obtain an improved version of KEManager for the 2010 course.

The second evaluation questionnaire was administered to students at the end of the course in June 2010. As the course unfortunately had a limited number of students, individual questions generally received only around 10 responses each. While this very low average response number dramatically limits the significance of the conclusions that can be drawn from the results obtained, the responses nevertheless offer a general idea of how the tool was perceived by users. Unlike the previous evaluation, the 2010 questionnaire included more complex questions. For these questions, students selecting a number from 1 to 5 corresponding to their level of agreement with the corresponding statement. All questions were categorized according to the following evaluation objectives:

- Correctness (C) of the tool.

- Completeness (X) of the tool.
- Effectiveness (E) of the tool.
- Usability (U) of the tool.

Additionally, questions were also grouped according to the following meta-tool aspects:

- General aspects (GE).
- User interface (UI).
- Knowledge acquisition (KA).
- Course project phases (CW).
- Graph modeling (GM).
- Printable representation (PR).

Tables 2, 3, 4, 5, 6, and 7 present the questionnaire results obtained for these six question groups. For each question, values are given for the average response score, standard deviation, and number of responses received.

Table 2. Questionnaire results about general aspects (GE) of KEManager.

Objective	Question	Average	Deviation	Answers
E	The tool helped me better understand knowledge engineering.	3.20	1.16	10
E	The tool helped me with the development of the course project.	3.60	1.28	10
U	The installation process was easy to perform.	4.20	0.97	10
X	The tool offered sufficient functionality for use in the subject.	4.20	0.97	10
U	Provided help and documentation allowed me to understand how the tool is used.	4.20	0.97	10

Of all questionnaire categories, the general aspects of the meta-tool resulted in the highest mean score. From this, the use of the framework implemented for the creation of a tool to help students better understand KE may be qualified as a

Table 3. Questionnaire results about user interface (UI) of KEManager.

Objective	Question	Average	Deviation	Answers
U	User interface was easy to understand and helped show how the tool works.	3.00	1.00	10
U	Graphic control layout was correct and helped understand the task performed.	2.88	0.73	9
U	Graphic control interaction was intuitive for getting and inserting data.	2.90	1.30	10
U	Window size did not affect the graphics control layout.	3.10	0.94	10

success. The high evaluation scores also given for knowledge acquisition indicate that students found the tool useful in the knowledge acquisition process. On the other hand, the course project phases (CW) and user interface received less positive scores. Indeed, phases have many more issues and aspects to be covered than knowledge acquisition techniques and, additionally, the integration of the former into KEManager with the appropriate user interface is much more difficult.

Tables 8 and 9 show the average values, sorted from highest to lowest, for each KEManager objective and aspect, respectively.

With regard to the tool objectives (see Table 8), although KEManager operability (Objective C) was considered, the evaluation results did not focus solely on this matter. The questionnaire also asked users to evaluate whether they felt that the meta-tool helped them work more quickly (Objective E) which, of course, is the principal objective of any KE tool. The highest average score was given for questions related to meta-tool completeness (Objective X), the drawback of other KE tools for which KEManager had been designed to correct. Therefore, it appears that the defined framework was useful in the way intended; namely, for the definition of combined methodologies. On the other hand, meta-tool usability (Objective U) was the lowest-scored objective, indicating the presence of flaws or deficiencies in the final GUI implementation. In fact, several student evaluators noted that the user interface defined for phases was difficult to understand.

Table 4. Questionnaire results about knowledge acquisition (KA) of KEManager.

Objective	Question	Average	Deviation	Answers
C	The knowledge acquisition process was properly integrated in the tool.	3.55	0.68	9
E	The knowledge acquisition process helped reduce the time spent on this part of the project.	3.77	1.03	
X	All aspects related to an acquisition session were complete and well-implemented.	3.70	0.64	10
X	There were enough acquisition techniques implemented for the course project.	3.88	0.87	9
C	Knowledge extraction was well-implemented and useful.	3.71	0.69	7
C	Open interview was well-implemented and useful.	3.57	0.72	7
C	Structured interview was well-implemented and useful.	3.57	0.49	7
C	Triadic method was well-implemented and useful.	3.57	0.49	7
C	Grid was well-implemented and useful.	4.11	0.73	9
E	The text analysis control was useful in the session analysis process.	3.71	0.69	7

With regard to aspects of KEManager (Table 9), all average evaluation results were located in the upper third of the scoring scale (i.e., between 3.33 and 5.00), indicating sufficient meta-tool functionality to work in KE. While general aspects of the meta-tool were scored very highly (3.93), the user interface received a low average score (2.97), indicating a need for improvement in order, in turn, to improve interaction with the user.

While the majority of free comments by evaluators focused on the GUI, many students also expressed gratitude for having been able to avoid the computations necessary in repertory grids and the aggregation of answers with weights in the harmonic average of problem identification.

These types of subjective benefits of KEManager are nonetheless difficult to prove. After all, there is a multiplicity of things that an individual might be able to do with KEManager that the individual might not be able to do without it.

Table 5. Questionnaire results about course project phases (CW) of KEManager.

Objective	Question	Average	Deviation	Answers
C	The problem identification phase was properly integrated in the tool.	3.66	0.81	9
E	The problem identification process helped reduce the time spent on this part of the project.	2.62	0.85	8
X	All aspects related to problem identification were complete and well-implemented.	3.50	0.70	8
E	Task evaluation helped me determine if KE was suitable for my project.	3.50	0.86	8
C	The conceptualization phase was properly integrated into the tool.	3.62	3.69	8
E	The conceptualization process helped reduce the time spent on this part of the project.	3.12	1.05	8
X	All aspects related to conceptualization were complete and well-implemented.	3.37	0.48	8
E	Integration with the CML2Editor tool helped me use CML2.	3.14	0.63	7

Despite the generality and non-task-specificity of the benefits obtained, support tools help users with the generation of documentation in interviews, computation in problem identification and repertory grids, completeness and randomness in the triadic method, and the overall coherence between the knowledge acquired and the knowledge represented.

7. Conclusions and Future Work

Results and analysis from this study show that the definition of a new methodology from a combination of different, pre-existing KE methodologies is possible through the application of the KEManager meta-tool. The meta-tool was created following the determination that the current state of the art for KE tools did not permit this possibility. The KE methodologies used here were those taught at the UC3M. The principal contributions of this study may be summarized as the following:

- A brief analysis of KE and the most important tools currently used with

Table 6. Questionnaire results about graphic modeling (GM) of KEManager.

Objective	Question	Average	Deviation	Answers
X	The diagram tool was complete and easy to use.	3.87	0.59	8
E	The diagram tool was complete and easy to use.	3.75	0.96	8
E	There were enough graphic components and pallets for the course project.	3.25	1.19	8
E	The possibility of defining attributes and relationships in certain components helped me better comprehend the modeling task, and helped reduce the time spent modeling.	3.62	0.85	8

Table 7. Questionnaire results about printable representation (PR) of KEManager.

Objective	Question	Average	Deviation	Answers
E	The possibility to export the project, or part of the project, to a printable representation helped me in the course project.	3.66	1.33	9

Table 8. Objective results.

Objective	Average
Completeness	3.68
Correctness	3.67
Effectiveness	3.40
Usability	3.38

it.

- A comprehensive description of the features of KEManager, the generic meta-tool developed here, which the user may define and use with different parts of any methodology's phases, and which includes several methods

Table 9. Tool aspect results.

Areas	Average
General aspects	3.93
Knowledge acquisition	3.71
Printable representation	3.66
Graphic modeling	3.62
Work phases	3.31
User interface	2.97

for the identification of a problem and the acquisition and modeling of knowledge.

- Results from student evaluations of KEManager performance in class projects offer valuable feedback about meta-tool weaknesses and suggestions about how these weaknesses may be eliminated.

Although the meta-tool possesses a particular strength (i.e., the ability to define and use a combined methodology with a good degree of completeness) not present in any other currently existing tool, student questionnaire responses we consider that it still has some indicate that meta-tool functionality could be increased even further with the elimination of particular weaknesses. The quality of the interaction between the meta-tool and user was identified by students as the greatest weakness to be addressed.

The methodology combined by the students integrates the problem identification evaluation method and knowledge acquisition techniques suggested by the authors of IDEAL, and the knowledge and communication models produced by the conceptualization phase of CommonKADS. While the combination used here was easy to execute, other combinations could be more complex and, therefore, produce interesting problems to solve in future studies. That said, as KEManager was defined to function as a generic tool that could combine any KE methodology, and in order to facilitate student comprehension of the two particular methodologies, no ad hoc features from IDEAL or CommonKADS were included in the GUI.

The questionnaire results considered here indicate not only the strengths and weaknesses of KEManager as a tool, but they also hint as how student evaluators felt about the problem identification and conceptualization phases, themselves. While it is not possible to obtain quantitative measurements about how the particular combined methodology phases facilitated the KE process, future studies could, as mentioned in the introduction, use KEManager with different methodological combinations, comparing and contrasting them with user evaluations (as done for the single combination defined from IDEAL and CommonKADS). A further benefit offered by KEManager that may be exploited in the future is its ability to facilitate

the indirect comparison of the relative strengths of certain parts of a methodology over others (although subjectively with user surveys).

While all minor technical problems reported by students have since been fixed, the current meta-tool may be improved/extended in four principal ways. Firstly, student questionnaires can be redesigned in order to obtain higher quality student feedback. The second way is related to how the tool is extended. One question that frequently arose during the implementation of the tool was whether KE phases and knowledge acquisition techniques could be defined as a composition of services, understood as the small functionality of the tool covering the GUI, model representation, and printing, rather than of methodologies. In other words, then, an extensible framework could be made with a finer granularity than a KE methodology. Thirdly, many other combinations of pre-existing KE methodologies can be developed in order to determine whether KEManager is generic enough to include them. Fourth and finally, the way the use of a combined KE methodology is evaluated (e.g., time taken, student final grades, or project size) could be automated, such that the contribution of KEManager to respond to KE difficulties could be appreciated more objectively.

In closing, this study has introduced a new meta-tool which presents an entirely unique strength: the ability to define and use combinations of KE methodologies. It is the authors' hope that the study prompts researchers and developers to view KE methodologies in a different light. While KEManager has, until now, been used exclusively for academic purposes at the UC3M and UNED, feedback on the use of KEManager in the industry would be welcome. In order to promote the free use of the tool (including its extension) not only for academic purposes, the software has been released on SourceForge and a website dedicated to the tool will also be launched.

Acknowledgements

Research by Fernando Fernández has been supported in part by the Spanish Ministry of Science and Innovation TIN2008-06701-C03-03 and TRA2009-0080 projects and by the Region of Madrid CCG10-UC3M/TIC-5597 project. Javier Carbó has been funded in part by the projects CICYT TIN2008-06742-C02-02/TSI, CICYT TEC2008-06732-C02-02/TEC, CAM CONTEXTS (S2009/TIC-1485), and DPS2008-07029-C02-02.

References

[1] T. R. Gruber, Formal Ontology in Conceptual Analysis and Knowledge Representation, from the book Towards Principles for the Design of Ontologies Used for Knowledge Sharing, Kluwer Academic Publishers, 1993.

[2] O. Corcho, M. Fernández-López, A. Gómez-Pérez. Methodologies, tools and languages for building ontologies: where is their meeting point?, *Data Knowl. Eng. 46, 1, July 2003*, pp. 41-64

[3] R. Studer, V.R. Benjamins and D. Fensel, Knowledge engineering: principles and methods *Data and knowledge engineering 25, 1-2*, 1998, pp. 161-197.
[4] G. Schreiber, H. Akkermans, A. Anjewierden, R. de Hoog, N. Shadbolt, W. Van de Velde, B. Wielinga, *The CommonKADS Methodology* (The MIT Press, 1999, ISBN: 0-262-19300-0).
[5] N. Juristo, J. Pazos, Towards a Joint Life Cycle for Software and Knowledge Engineering, *Knowledge Oriented Software Desing*, Amsterdam, 1993.
[6] F. Alonso Amo, N. Juristo, J. Pazos. Trends in Life- Cycle Models for SE and KE: Proposal for a Spiral-Conical Life-Cycle Approach. *International Joumal of Software Engineering and Knowledge Engineering. Vol. 5. No. 3*, 1995.
[7] F. Alonso Amo, J. L. Fuertes, L. A. Martinez, C. Montes. A Knowledge Engineering Software Development Methodology Applied to a SpirdConical Life Cycle. *Proceedings of the 9 International Conference on Software Engineering and Knowledge Engineering. SEKE97*, 1997, pp. 32-37.
[8] M. Fernandez-Lopez, A. Gómez-Pérez. Overview and analysis of methodologies for building ontologies, *Knowl. Eng. Rev. 17, 2*, June 2002, pp. 129-156
[9] E. Downs, P. Clare, I. Coe, *Structured Systems Analysis and Design Method*(Prentice Hall, 1988, ISBN: 0138536988).
[10] P. Harmon, D. King, *Expert Systems* (John Wiley and Sons, Inc, 1985).
[11] D. A. Waterman, *A Guide to Expert Systems* (Addison-Wesley, Masachusets, USA, 1986).
[12] Taki, I, Expert Model for Knowledge Acquisition in the ICOT, *Technical Presentation at Computer Science Department*, Carnegie Mellon University, Pittsburgh, PA, 1986.
[13] M. A. Carrizo, J. E. Girad, J. P. Jones, *Building Knowledge Systems: Developing and Managing Rule-Based Applications*, McGraw-Hill, 1989.
[14] R. Alberico, M. Micco, *Expert Systems for reference and information retrieval*, Mockler Corporation, IK, 1990.
[15] J. S. Edwars, *Building Knowledge-Based Systems. Towards a Methodology*, 1991.
[16] G. Schreiber, B. Wielinga, J. Breuker, *KADS: a principled approach to knowledge-based system development* (Academic Press, Inc, 1993, ISBN: 0-12-629040-7).
[17] http://web1.eng.coventry.ac.uk/moka/default.htm
[18] Z. Zheng, W. Li, An Introduction to KEDE - A Hybrid Knowledge Engineering Development Environment, *Proc. of the 2nd International IEEE Conf. on Tools for Artificial Intelligence*, 1990, pp. 63-69.
[19] M. Huneault, C. Rosu, R. Manoliu, F. D. Galiana, A Study of Knowledge Engineering Tools In Power Engineering Applications, *IEEE Transactions on Power Systems, Vol. 9 No. 4*, 1994.
[20] J. Hyun, P. Hyun, An integrated Knowledge Base Development Tool for Knowledge Acquisition and Verification for NPP Dynamic Alarm Processing Systems, *Annals of Nuclear Energy 29*, 2002, pp. 447-463.
[21] A. J. Duineveld, R. Stoter, M. R. Weiden, B. Kenepa and V. R. Benjamins, WonderTools? A Comparative Study of Ontological Engineering Tools, *Int. J. Human-Computer Studies 52*, 2002, pp. 1111-1133.
[22] M. A. Musen, R. W. Fergerson, W. E. Grosso, N. F. Noy, M. Crubzy, J. H. Gennari, Component-Based Support for building Knowledge-Acquisition Systems, in *Proc. of the Conf. on Intelligent Information Processing of the International Federation for Information Processing Word Computer Congress*, 2000, pp. 18-24.
[23] R. M. Simpson, T. L. McCluskey, W. Zhao, R. S. Aylett, C. Doniat, An integrated Graphical Tool to support Knowledge Engineering in AI Planning, —, 2001.
[24] H. Richter, G. Abowd, C. Miller, H. Funk, Tagging Knowledge Acquisition Sessions

To Facilitate Knowledge Traceability, *International Journal of Software Engineering and Knowledge Engineering Vol. 14, No 1*, 2004, pp. 3-19.

[25] C. W. Chan, From Knowledge Modeling to Ontology Construction, *International Journal of Software Engineering and Knowledge Engineering Vol. 14, No 6*, 2004, pp. 603-624.

[26] T. C. D'Agostini, H. C. Hoeschl, A. Bortolon, E. Mattos, C. Souza, Knowledge Engineering Suite: A Tool to Create Ontologies for Automatic Knowledge Representation in Knowledge-Based Systems, *Lecture Notes in Computer Science Vol. 3591/2005*, 2005, pp. 249-260.

[27] Anders K. Ericsson, James J. Stasewski, Chapter 9: Skilled Memory and Expertise: Mechanisms of Exceptional Performance. In David Klahr and Kenneth Kotovsky. *Complex Information Processing: The Impact of Herbert A. Simon.* Hillesdale N.J.: Lawrence Erlbaum Associates, 1989.

[28] R. Pressman, *Software Engineering: A practitioner's aproax*(Pressman, 2005, ISBN: 0072853182).

[29] http://repgrid.com/

[30] http://epistemics.co.uk/Notes/55-0-0.htm

[31] http://protege.stanford.edu/

[32] http://kemanager.sourceforge.net/

[33] A. Alonso, B. Guijarro, A.Lozano, J. Palma, M. J. Taboada, *Ingeniería del Conocimiento: Aspectos Metodológicos*(Pearson - Prentice Hall, 2004, ISBN: 84-205-4192-3).

[34] A. Gómez, N. Juristo, C. Montes, J. Pazos, *Ingeniería del Conocimiento* (Editorial Centro de Estudios Ramón Areces, 1997, ISBN: 84-8004-269-9).