KNOWLEDGE ENGINEERING USING THE UML PROFILE
Adopting the Model-Driven Architecture for Knowledge-Based System Development

Mohd Syazwan Abdullah [1,2], Richard Paige[2], Ian Benest[2] and Chris Kimble[2]
[1]*Faculty of Information Technology, Universiti Utara Malaysia, 06010 Sintok, Kedah, Malaysia*

[2]*Department of Computer Science, University of York, Heslington, York, United Kingdom*

Keywords: Knowledge engineering, knowledge modelling, knowledge-based system, UML profile.

Abstract: Knowledge engineering (KE) activities are essential to the process of building intelligent systems; it conceptual modelling is exploited so that the problem-solving techniques used may be understood. This paper discusses platform independent conceptual modelling of a knowledge intensive application, focusing on knowledge-based systems (actually, a rule-based KBS) in the context of a model-driven architecture (MDA). It emphasises the use of problem-solving methods for developing the knowledge-level models. An extension to the Unified Modeling Language (UML), using its profile extension mechanism, is presented. The profile discussed in this paper has been successfully implemented in the eXecutable Modelling Framework (XMF) – a Meta-Object-Facility (MOF) based UML tool. A case study demonstrates the use of this profile; the prototype is implemented in the Java Expert System Shell (Jess).

1 INTRODUCTION

Knowledge-based systems (KBS) were developed for managing codified knowledge (explicit knowledge) in Artificial Intelligence (AI) systems (Giarratano and Riley, 2004). These were known as expert systems and were originally created to emulate human expert reasoning (Studer *et al*., 1998). They have been one of the most successful outcomes from AI research (Metaxiotis and Psarras, 2003) and have been adopted in the medical, business, manufacturing, and other domains.

KBS are developed using knowledge engineering (KE) techniques (Studer *et al*., 1998), that are similar to those used in software engineering (SE), but they emphasize knowledge rather than data or information processing. Central to this is the conceptual modelling of the system during the analysis and design stages of the development (known as knowledge modelling). A number of KE methodologies have emphasized the use of models, for example: CommonKADS, MIKE, KARL and others (Gomez-Perez and Benjamins, 1999). KBS continue to evolve as the need to have a stable technology for managing knowledge grows; its current role as an enabler in knowledge management initiatives has led to its wider acceptance (Ergazakis *et al*., 2005). It has matured from a non-scalable technology (Giarratano and Riley, 2004).Once restricted to the research laboratory, it is now used for demanding commercial applications and is a tool widely accepted by industry (Liebowtiz, 2001). As a result, the Object Management Group (OMG), which governs object-oriented software modelling standards, has started the standardisation process for production rule representation (PRR) (OMG, 2003) and knowledge-based engineering (KBE) services (OMG, 2004). Standardising PRR is vital as it will allow interoperability of rules between different inference engines – much needed by industry (McClintock, 2005, Krovvidy *et al*., 2005).

This paper is organised thus. Section 2 discusses knowledge modelling. Section 3 explains the rationale for having an extension in UML. Section 4 overviews the UML extension mechanism, and section 5 describes the knowledge modelling profile. With the aid of a case study, section 6 illustrates how the profile can be used to develop a KBS. Section 7 concludes with directions for future work.

2 KNOWLEDGE MODELLING

Newell (Newell, 1982) emphasises the importance of developing problem-solving models of the domain rather than focusing on knowledge representation. This is usually referred to as the knowledge-level principle and differs from the previous mining view (Motta, 2002). Two strands of research have been established based on this knowledge-level modelling principle (Chan, 2004).

One emphasises the refinement of existing knowledge-level principle formalisation languages such as KARL (Angele *et al*., 1996) and KADS's ML2 language (Flores-Mendez *et al*., 1998). The other area deals with developing knowledge-level models for a variety of tasks and domains in order to understand the problem-solving techniques used. There are two distinct Knowledge Modelling (KM) approaches: the problem-solving method (PSM), and the domain ontology (Chan, 2004, Schreiber *et al*., 1999, Angele et al., 1996, Dieste *et al*., 2002). The PSM exploits domain independent abstract models that describe the generic inference patterns for different tasks (Angele *et al*., 1996); while an ontology defines the commonly agreed vocabularies for representing the domain knowledge (Gruber, 1993). The focus of this paper is on using PSM techniques for developing knowledge-level models.

While it is commonly agreed that conceptual modelling is an important stage in any software system construction (Naumenko and Wegmann, 2002), both SE and KE communities have developed different modelling techniques that are almost unrelated (Dieste *et al*., 2002) resulting in fundamental computational differences in the way they solve the same problem (Juristo, 1998). This makes it difficult to interchange their models (Juristo, 1998). Nevertheless, most KE modelling notations have adopted those from the more established SE domain (though KE contributed object-oriented development through frames). Among the approaches to KE, CommonKADS is the most comprehensive and well structured methodology (Motta, 2002) and is widely used. Its graphical notation has a strong resemblance to those used in object languages such as the Unified Modeling Language (UML) (Dieste *et al*., 2002).

3 RATIONALE FOR EXTENSION

Research has shown that neither technical nor economic factors determine whether KBS technology will be successfully adopted, but rather it is the organisational and managerial environment that is the main determinant (Gill, 1995, Tsui, 2005). Gill (1995) highlights one of the problems: the management of the development team. KBS projects are specialised in nature requiring team members to have knowledge of both the problem domain and the development tools. As a result, the team members are skilful individuals and the success of the project is threatened if one or more leave the team mid-way through the development or during the maintenance period. But a KBS that is designed using an appropriate, well-understood, standard language for conceptual modelling along with a methodologically sound representation technique, should be readily understood by new team members.

The major problem with KM is that there is no standard language available to model the knowledge for developing a KBS (Chan, 2004, McClintock, 2005, Krovvidy *et al*., 2005). Most of the languages used are adapted from SE. The languages used in KM are project based using a mix of notations such as UML, IDEF, SADT etc. The SE community has adopted UML as the *de facto* standard for modelling object-oriented systems and the KE community should do the same. This would be beneficial in the long-term as KBS can be easily integrated into other enterprise systems (Krovvidy *et al*., 2005, Giarratano & Riley, 2004) particularly if their designs were based on a standard language; it would help facilitate communication and sharing of blueprints among developers (Abdullah *et al*, 2002). OMG's PRR (OMG, 2003) should go some way to satisfy this standardisation requirement.

The motivation for the extension was to accommodate UML for KM in designing KBS. There was a need for a standardised approach in designing KBS, and reaping the benefits of using UML (better tool support, large user base familiar with the language, and an evolving standard).

4 THE UML EXTENSIONS

The OMG's Model Driven Architecture (MDA) – a model-driven engineering framework – provides integration with, and interoperability between, different models developed using its standards (Muller *et al*., 2003) (such as UML, Meta-Object Facility (MOF), and others). The growth of MDA will fuel the demand for more meta-models to cater for domain specific modelling requirements.

Profiles have 'precisely' defined semantics and syntax, which enables them to be formally integrated into UML, though of course they must adhere to the profile requirements proposed by OMG. Previous profile development for modelling knowledge has concentrated only on certain task types such as product design and product configuration. In contrast, the work described here emphasises the development of a generic profile.

Developing a meta-model for KM will enable it to be integrated into the MDA space allowing the relation between the knowledge models and other language models to be understood. It provides for seamless integration of different models in different applications within an enterprise.

UML is a general-purpose modelling language (Muller *et al*., 2003) that may be used in a wide range of application domains. It can be extended to model domains that it does not currently support, by extending the modelling features of the language in a controlled and systematic fashion. The OMG (OMG, 2001) defines two mechanisms for extending UML: profiles and meta-model extensions. Both extensions have (unfortunately) been called profiles (Muller *et al*., 2003).

The "lightweight" extension mechanism of UML (OMG, 1999) is the profile. It contains a pre-defined set of Stereotypes, TaggedValues, Constraints, and notation icons that collectively specialize and tailor the UML. The main construct in the profile is the stereotype that is purely an extension mechanism. In the model, it is marked as «stereotype» and has the same structure (attributes, associations, operations) as that defined by the meta-model. However, the usage of stereotypes is restricted; changes in the semantics, structure, and the introduction of new concepts to the meta-model are not permitted (Perez-Martinez, 2003).

The "heavyweight" extension mechanism for UML (known as the meta-model extension) is defined through the Meta-Object Facility (MOF) specification (OMG, 2002) which involves the process of defining a new meta-model (Perez-Martinez, 2003). This approach should be favoured if the semantic gap between the core modelling elements of UML and the newly defined modelling elements is significant (Muller *et al*., 2003).

It is preferable to create a profile using the "lightweight" extension since it is easier to use, easier to introduce new concepts through the existing meta-model and has better tool support compared with that of the meta-model extension. The work presented in this paper exploits the "lightweight" extension using the XMF approach.

5 THE KM PROFILE

The OMG UML specifications only specifies what profiles should constitute and not how to design them. By adopting the XMF (eXecutable Meta-modelling Framework) approach (Clark *et al*., 2005), the profile development is structured into well-defined stages that are easy to follow and methodologically sound. The XMF is a newly developed object-oriented meta-modelling language, and is an extension to existing standards for meta-models such as MOF and UML. XMF offers an alternative approach in profile design, which allows modification, or addition, of new modelling constructs; and these are easily integrated into the core meta-model of UML. The creation of a profile can be divided into three steps: the derivation of an abstract syntax model of the profile concepts, a description of the profile's semantics, and the presentation of the profile's concrete syntax (not discussed here) if this is different from UML diagrams.

5.1 Abstract Syntax

The abstract syntax describes the vocabulary of concepts in the profile and the associations between those concepts. It also defines the well-formed-ness rules that determine the models validity. The processes involved in creating the abstract syntax are: identifying the domain specific concepts to be modelled including the related well-formed-ness rules for constraining the manner in which the concepts may be used; modelling the concepts by creating an abstract syntax meta-model of the profile; defining the well-formed-ness rules of the profile; defining operations and queries related to the profile where applicable and validating and testing the profile to ensure the correctness of the abstract syntax model.

Profile Concept

The concepts for the profile are re-used from the existing BNF definition of the CommonKADS Conceptual Modelling Language (CML) (Schreiber *et al*., 1999); this provides a well-defined and well-established main set of concepts for the domain. Most of these elements are generally adopted in the KBS literature and are widely used for representing concepts in KBS in the KE domain. These concepts are itemised in Table 1 and the abstract syntax model of the profile is shown in Figure 1.

Model Extension

The knowledge modelling profile concept extends the existing meta-models of UML by defining the profile's abstract syntax. There are three places where the profile can be viewed as an extension to UML and these are: Class, Named Element and Constraints, all of which are central to the core UML meta-model and are also found in UML. The KM concept class enables the concept to inherit all the features of a class and allows it to specify attributes and constraints on the attribute values. Other concepts such as inference, task, task method, dynamic role, static role, and the transfer function are also viewed as a subclass of an UML Class and inherit the class features. This allows operations related to objects to be expressed, such as an execute inference call from the task method, the execution of the inference process and the access to knowledge in the knowledge base through the static role. At the same time, it allows these elements to specify attributes.

Table 1: Main Knowledge Modelling Concepts.

Modelling Concept	Description
Concept (class)	Class that represents the category of things
FactBase/Working Memory	Collection of information/facts that will be matched against the rule
Inference	The lowest level of functional decomposition into primitive reasoning steps
Transfer Function	Transfers information between the reasoning agent and external entities (system, user)
Task	Defines the reasoning function
Task Method	Describes the realization of the task through sub-function decomposition
Static Knowledge Role	Specifies the collection of domain knowledge that is used to make the inference
Dynamic Knowledge Role	Run-time inputs and outputs of inferences
Rule Type	Categorization and specification of knowledge
Rule	Expressions that involve an attribute value of a concept
Knowledge Base	Collection of data stores that contain instances of domain knowledge types

Knowledge base is a subclass of the UML class. It has a 'content' slot for specifying tables. This is a natural choice for a subclass as the knowledge base is actually a collection of tables grouped together in order to store rule type instances. The profile's tuple concept is also extended from Class. Constraint class is a subclass of the UML meta-model, incorporating profile concepts such as axioms and rule type expressions. All these concepts need the ability to express constraints; this class allows for this. Rule Type is a subclass of the UML Named Element, allowing rules to be identified using a name. All the associations described in the profile are extensions of the UML association class. However, they are not shown in the profile, as it would clutter the diagram.

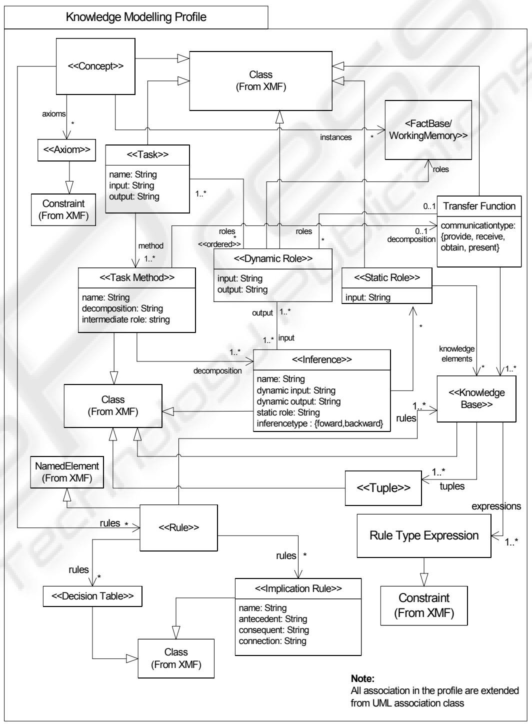

Figure 1: Knowledge Modelling Profile.

Well-formed-ness Rules

Defining the well-formed-ness rules of the profile modelling elements in OCL helps to make impossible illegal models that might otherwise be created using the profile concepts. The following well-formed-ness rules are defined for the concepts in the profile and listed in Table 2. An example of one of the rules written in XMF OCL syntax) is as follows (each inference must have a unique name):

```
context Inference
@Constraints InferencesHaveUniqueNames
inference->forAll (s1 | states->forAll
(s2 | s1.name = s2.name implies s1 =
s2))end
```

Table 2: Profile Well-formed-ness rule.

Class	**Well-formed-ness rule description**
Concept	• Concept cannot own operations.
Inference	• Inference must have a unique name. • Inference can only be associated with task method, dynamic and static role.
Transfer Function	• Transfer function can only be associated with task method and dynamic role.
Task	• Task must have a unique name. • Task can only be associated with task method.
Task Method	• Task method must have a unique name. • A task method can only be associated with transfer function and inference. • Task method can only be decomposed to task, inference and transfer function.
Factbase	• Factbase can only be associated with concept and dynamic role.
Static Knowledge Role	• Static role can only be associated with inference and knowledge-base.
Dynamic Knowledge Role	• Dynamic role can only be associated with factbase, transfer function and inference
Rule Type	• Any one of the rule types must exist: constraint, implication and decision table.
Implication Rule	• Rule can only be associated with concept and knowledge-base.
Knowledge Base	• Knowledge base can only be associated with tuple, rule and static role.
All associations	• Can only be used to join those concepts that it is linked with in the profile

5.2 Semantics

The semantics describe the meanings of the concepts within the profile in terms of behaviour, static properties or the means by which it may be translated into another language. Semantics are important to the profile as they are used to communicate the meaning of the models amongst its users and avoid misinterpretation; they are a core part of the profile's meta-model and are there instead of formal (mathematical) methods that are often difficult to comprehend.

The Dynamic Role class specifies the 'information' flow of attribute instances from the concepts. It also specifies the outputs that arise from executing the inference sets. The output of this inference process is the 'result' of matching the antecedent of the rule with the consequent part. Depending on what the knowledge-based system is reasoning about, if it is not the final output of the system, then the output can be used in another inference.

The Static Role class is the function responsible for fetching the collection of domain knowledge (rules) from the knowledge base prior to an active inference. Inferences do not access the knowledge base directly, but request the necessary rules related to the particular inference from the static roles. In some knowledge-based system shells this is similar to posting the rules to the inference process or similar to setting which rule should be fired. This allows the inference process to handle a specific reasoning task and invoke those rules that are appropriate.

An Inference process class executes a set of algorithms for determining the order in which a series of non-procedural, declarative statements are to be executed. The inference process infers new knowledge from information/facts that are already known. The Task Method invokes this. The input (information/fact) used by this process is provided by the dynamic role. The result of the inference process is then passed to the dynamic role. The knowledge element used in the inference is accessed through the Static Role, which fetches the group of rules from the knowledge base. There are several different inference processes for a given task, most of which are run in the background by the inference engine.

The Rule class of the profile describes the modelling of rules within the domain concept. Rule class is used to represent knowledge elements in KBS and is viewed as 'information about information'. Rule class allows for rules to be in different formats. There are three types of rule: implication rule, decision table and constraint rule. An implication rule is of the form: 'if-then' premise followed by an action. This type of representation is widely used in KBS; they are known as production rules. A decision table is an addition to the rule class. It is introduced here because certain rules are best expressed in the form of a decision table, even though they are usually converted to flattened production rules. This paper only concentrates on rule-based KBS as it is the one widely adopted by industry.

The knowledge base class contains domain knowledge, represented as rules, which are used by the inference process. The contents of the knowledge base are organized in tuples (records). A tuple is used to group rules according to their features. This allows the partitioning of the knowledge base into modules that enables the inference process to access the rules faster. The maintainability of the rules is enhanced when it is organised in this manner.

6 CASE STUDY – CLINICAL PRACTICE GUIDELINE RECOMMENDATIONS

The Clinical Practice Guideline (CPG) Recommendations contains statements that are graded according to the following three strengths of evidence: (a) generally consistent findings in a majority of multiply acceptable studies; (b) either based on a single acceptable study, or weak or inconsistent findings in multiply acceptable studies; (c) limited scientific evidence that does not meet all the criteria of acceptable studies of good quality. The recommendations cover assessment of leg ulcers, management of venous leg ulcers, cleansing, removal of medical debris, dressing and contact sensitivity, education and training, and quality assurance categories. A KBS for educational purposes was designed to list the recommendations based on (a) evidence strength; (b) evidence strength and category; (c) category only. Figure 2 shows how the profile has been used to represent part of the CPG case study.

The profile here only concentrates on showing the task of making recommendations (*considered as classification task-type*), based on the user-selected criteria. The task is invoked by the task method "prune set" which is executed by several inferences and intermediate roles. For the matching process to provide recommendations, different sets of rules are used depending on the criteria selected by the user. To arrive at a recommendation, the inference would need the pertinent knowledge or rules from the knowledge base. This is provided by the static role, and the facts (CPG recommendations in the factbase) to match them are gathered by the dynamic role.

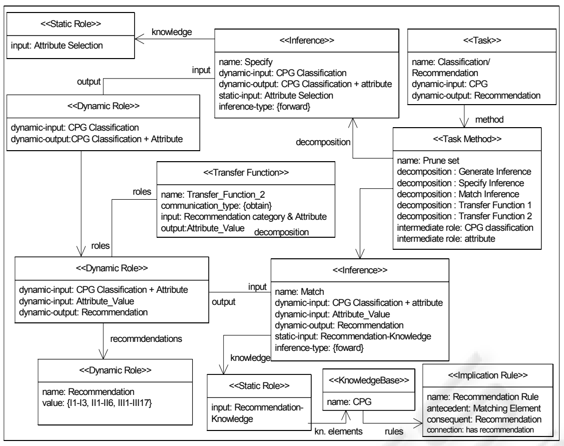
<<Static Role>>
input: Attribute Selection
knowledge
<<Inference>>
name: Specify
dynamic-input: CPG Classification
dynamic-output: CPG Classification + attribute
static-input: Attribute Selection
inference-type: {forward}
<<Task>>
name: Classification/ Recommendation
dynamic-input: CPG
dynamic-output: Recommendation
input
output
<<Dynamic Role>>
dynamic-input: CPG Classification
dynamic-output:CPG Classification + Attribute
method
decomposition
<<Task Method>>
name: Prune set
decomposition : Generate Inference
decomposition : Specify Inference
decomposition : Match Inference
decomposition : Transfer Function 1
decomposition : Transfer Function 2
intermediate role: CPG classification
intermediate role: attribute
<<Transfer Function>>
name: Transfer_Function_2
communication_type: {obtain}
input: Recommendation category & Attribute
output:Attribute_Value
decomposition
roles
roles
<<Dynamic Role>>
dynamic-input: CPG Classification + Attribute
dynamic-input: Attribute_Value
dynamic-output: Recommendation
input
output
<<Inference>>
name: Match
dynamic-input: CPG Classification + attribute
dynamic-input: Attribute_Value
dynamic-output: Recommendation
static-input: Recommendation-Knowledge
inference-type: {foward}
recommdendations
knowledge
<<Dynamic Role>>
name: Recommendation
value: {I1-I3, II1-II6, III1-III17}
<<Static Role>>
input: Recommendation-Knowledge
<<KnowledgeBase>>
name: CPG
<<Implication Rule>>
name: Recommendation Rule
antecedent: Matching Element
consequent: Recommendation
connection: has recommendation
kn. elements
rules

Figure 2: CPG Recommendations Case Study Model.

The case study was implemented as a prototype system in the Java Expert System Shell (Jess), that is based on the popular CLIPS program (Friedman-Hill, 2003). Because of the declarative nature of expert system shell programming, the concepts of the profile cannot be entirely matched to a Jess meta-model. However, the KM profile was very useful in understanding the KBS requirements for the CPG recommendations. Some sample rules for listing recommendations based on evidence strength (in the actual recommendation each recommendation has a brief explanation rather than ID shown below as I1, II2, III4, etc.) are:

If evidence.strength = I Then Recommendation = {I1, I2, I3, I4}
If evidence.strength = II Then Recommendation = {II1, II2, II3, II4, II5, II6}
If evidence.strength = III Then Recommendation = {III1, III2, III3, III4, III5,…. to III19}

This rule set is mapped into Jess code as follows:

```
( defrule strength-I
   ( user (strength ?i&:(= ?i 1)))
  => assert  (recommendation I1 , I2 ,
I3 , I4)
explanation "Strength equals 1"))))

( defrule f strength-II
   ( user (strength ?i&:(= ?i 2)))
  => (assert  (recommendation II1 , II2
, II3 , II4 ,II5 ,II6)
  explanation "Strength equals 2"))))

( defrule strength-III
   ( user (strength ?i&:(= ?i 3)))
  => (assert  (recommendation III1 ,
III2 , III3 , III4 ,III5 ,III6 ,III7,
III8,III9,III10,III11,III12,III13,III14
,III15,III16,III17, III18, III19)
explanation "Strength equals 3 "))))
```

7 CONCLUSION AND FUTURE WORK

KBS development is similar to that experience in software engineering; both rely on conceptual modelling of the problem domain to provide an orientation as to how the system addresses the problem. UML has been adopted by those working in the SE domain as a standard for modelling, but there is still no consensus in the KE domain. This paper describes an extension to UML using the (lightweight) profile mechanism for knowledge modelling that allows KBS to be designed using an object-oriented approach. The profile has been successfully tested on several case studies. This includes designs from scratch and re-engineering existing KBS. Currently work has concentrated on building an Eclipse plug-in to support the profile. The plug-in will allow profile-compliant diagrams to be drawn and validated, and XML or XMI representations produced. The infrastructure in the Eclipse plug-in will make this mapping straightforward to implement.

The future work in this area involves studying how to map the profile to a specific inference engine meta-model and work in this area is in progress (Wu, 2004). Jess will be used initially as this has been widely adopted and will help assess not only the utility of the profile for building realistic KBS, but also the utility of XMF for capturing the meta-models and building the transformations.

REFERENCES

Abdullah, M.S., Benest, I., Evans, A., & Kimble, C. (2002) Knowledge Modelling Techniques for Developing Knowledge Management Systems. *In Proceedings of the 3rd European Conference on Knowledge Management*, Dublin, Ireland.

Angele, J., Decker, S., Perkuhn, R. & Studer, R. (1996) Modeling Problem-Solving Methods In New Karl. In *Proceedings Of Tenth Knowledge Acquisition For Knowledge-Based Systems Workshop (Kaw'96).* Calgary, Canada.

Chan, C. W. (2004) Knowledge And Software Modeling Using UML. *Software And Systems Modelling,* 3**,** 294-302.

Clark, T., Evans, A., Sammut, P. & Willians, J. (2005) *Metamodelling For Model-Driven Development:* To Appear.

Dieste, O., Juristo, N., Moreno, A. M., Pazos, J. & Sierra, A. (2002) Conceptual Modelling In Software Engineering And Knowledge Engineering: Concepts, Techniques And Trends. In Chang, S. K. (Ed.) *Handbook Of Software Engineering & Knowledge Engineering.* World Scientific Publishing Company.

Ergazakis, K., Karnezis, K., Metaxiotis, K. & Psarras, I. (2005) Knowledge Management In Enterprises: A Research Agenda. *Intelligent Systems In Accounting, Finance And Management,* 13**,** 17-26.

Flores-Mendez, R. A., Van Leeuwen, P. & Lukose, D. (1998) Modeling Expertise Using Kads And Model-Ecs. In *Eleventh Workshop On Knowledge Acquisition, Modeling And Management.* Banff, Canada.

Friedman-Hill, E. (2003) *Jess In Action: Rule-Based System In Java,* Greenwich, US, Manning Publication.

Giarratano, J. C. & Riley, G. D. (2004) *Expert Systems: Principles And Programming,* Boston, Massachusetts, Course Technology.

Gill, G. T. (1995) Early Expert Systems: Where Are They Now? *MIS Quarterly,* 19**,** 51-81.

Gomez-Perez, A. & Benjamins, V. R. (1999) Overview Of Knowledge Sharing And Reuse Components: Ontologies And Problem-Solving Methods. In *IJCAI-99 Workshop On Ontologies And Problem-Solving Methods (KRR5).* Stockholm, Sweden.

Gruber, T. R. (1993) Toward Principles For The Design Of Ontologies Used For Knowledge Sharing. Stanford University. Report No. KSL-93-04.

Juristo, N. (1998) Guest Editor's View. *Knowledge Based System,* 11**,** 77-85.

Krovvidy, S., Bhogaraju, P. & Mae, F. (2005) Interoperability And Rule Languages. *W3C Workshop On Rule Languages For Interoperability.* Washington, D.C., USA.

Liebowtiz, J. (2001) If You Are A Dog Lover, Build Expert System; If You Are A Cat Lover, Build Neural Networks. *Expert System With Applications,* 21**,** 63.

McClintock, C. (2005) Ilog's Position On Rule Languages For Interoperability. *W3C Workshop On Rule Languages For Interoperability.* Washington, D.C., USA.

Metaxiotis, K. & Psarras, J. (2003) Expert Systems In Business: Applications And Future Directions For The Operations Researcher. *Industrial Management And Data System,* 103**,** 361-368.

Motta, E. (2002) The Knowledge Modelling Paradigm In Knowledge Engineering. In Chang, S. K. (Ed.) *Handbook Of Software Engineering & Knowledge Engineering.* World Scientific Publishing.

Muller, P.-A., Studer, P. & Bezivin, J. (2003) Platform Independent Web Application Modeling. In Stevens, P., Whittle, J. & Boochgrady. (Eds.) *The Sixth*

International Conference On The Unified Modeling Language (Uml 2003). Springer-Verlag.

Naumenko, A. & Wegmann, A. (2002) A Metamodel For The Unified Modeling Language. In Jezequel, J. M., Hussmann, H. & Cook, S. (Eds.) *UML 2002.* Dresden, Germany, Springer-Verlag Berlin.

Newell, A. (1982) The Knowledge Level. *Artificial Intelligence,* 18**,** 87-127.

OMG (1999) Requirements For UML Profile. Framingham, MA, U.S.A., Object Management Group.

OMG (2001) Unified Modeling Language Specification (Version 1.4).

OMG (2002) MOF Specification Version 1.4.

OMG (2003) Production Rule Representation - Request For Proposal. Object Management Group.

OMG (2004) KBE Services For Engineering Design - Request For Proposal. Object Management Group.

Perez-Martinez, J. E. (2003) Heavyweight Extensions To The UML Metamodel To Describe The C3 Architectural Style. *ACM Sigsoft Software Engineering Notes,* 28.

Schreiber, G., Akkermans, H., Anjewierden, A., De Hoog, R., Shadbolt, N., De Velde, W. & Wielinga, B. (1999) *Knowledge Engineering And Management: The Commonkads Methodology,* Massachusetts, MIT Press.

Studer, R., Benjamins, R. & Fensel, D. (1998) Knowledge Engineering: Principles And Methods. *Data & Knowledge Engineering,* 25**,** 161-197.

Tsui, E. (2005) The Role Of It In KM: Where Are We Now And Where Are We Heading. *Knowledge Management,* 9**,** 3-6.

Wu, C.G. (2004) Modelling Rule-Based Systems with EMF. Accessed at http://www.eclipse.org/articles/