Proceedings of the 2004 IEEE
Conference on Cybernetics and Intelligent Systems
Singapore, 1-3 December, 2004

Modelling Knowledge Based Systems Using The eXecutable Modelling Framework (XMF)

Mohd Syazwan Abdullah[1,2], Andy Evans[1]
[1]Department of Computer Science
University of York, Heslington, YO10 5DD
York, United Kingdom
syazwan, andye, idb, paige, kimble @cs.york.ac.uk

Ian Benest[1], Richard Paige[1], Chris Kimble[1]
[2]Faculty of Information Technology
Universiti Utara Malaysia
06010 Sintok, Kedah DA, Malaysia
pathma @webmail.uum.edu.my

***Abstract*—There is no standardised approach to modelling knowledge-based systems; where modelling is adopted, the techniques used are those from the software engineering domain. These tend to be used in an *ad hoc* way and are highly dependent on the experience of the knowledge engineers. This paper presents the adoption of a profile mechanism for the design of knowledge-based systems. The profile is created using the meta-model extension approach of UML and is based on XMF (eXecutable Meta-modelling Framework). XMF is an extension to the existing standards for meta-models: MOF, OCL and QVT. XMF offers an alternative approach in profile design which allows modification or the addition of new modelling constructs that are easily integrated with the core meta-model of UML.**

Keywords—knowledge-based system; XMF Profile; knowledge modelling; executable models;

I. INTRODUCTION

Knowledge-based systems (KBS) are developed using knowledge engineering (KE) techniques [1], which are similar to those used in software engineering (SE), but have an emphasis on knowledge rather than on data or information processing. As such, they inherently advocate an engineering approach to the process of developing a KBS. Central to this process is the conceptual modelling of the system during the analysis and design stages of the development process. And many knowledge engineering methodologies have been developed with an emphasis on the use of models, for example: CommonKADS [2], MIKE [3], Protégé [4], and KARL [3].

In first generation expert systems, the knowledge of the expert (or experts) was captured and translated into a set of rules. This was essentially, a process of knowledge transfer [3]. The disadvantage of this approach is that the captured knowledge in the form of hard-coded rules within the system provides little understanding of how the rules are linked or connected with each other [2]. As a result, when the knowledge base needs updating, there is a substantial effort required to ensure that the knowledge base remains correct. KE is no longer simply a means of mining the knowledge from the expert's head [2]. It now encompasses *"methods and techniques for knowledge acquisition, modelling, representation and use of knowledge"* [2].

This paper demonstrates a systematic approach to modelling and designing KBSs in a purely object-oriented fashion through the use of profile mechanism. The novelty of the system design lies in the profile that is used to create it. The profile is constructed using compliant standards of modelling software systems by adopting the XMF approach. XMF provides tool support for designing and verifying models as well as executing the models. It is one of the latest techniques in modelling and this work demonstrates the use of this approach.

This paper is organised as follows: Section II describes and discusses the KBS design process and the use of conceptual modelling. Section III gives an overview of the Unified Modeling Language (UML) and profile extension mechanism. Section IV explains the profile design process using the XMF approach, while Section V illustrates how the KBS Modelling Profile can be used as part of the development of a KBS. Section VI concludes, indicating the direction for future work.

II. KNOWLEDGE-BASE SYSTEM DESIGN

Knowledge engineering is no longer simply a means of mining the expert's understanding and appreciation of a domain of knowledge [2]. It now encompasses *"methods and techniques for knowledge acquisition, modelling, representation and use of knowledge"* [2]. Schreiber *et al* [2] argue that models are important for understanding the working mechanisms within a KBS; such mechanisms are: the tasks, methods, how knowledge is inferred, the domain knowledge and its schemas. A further benefit arising from the shift towards the modelling approach is that fragments of knowledge may be re-used in different areas of the same domain [3] making systems development faster and more efficient. In the past, most knowledge systems had to be developed afresh each time a new system was needed, and it could not interact with other systems in the organization. So the paradigm shift towards a modelling strategy has resulted in reducing development costs [2].

Although a KBS is developed using knowledge engineering techniques, the modelling aspects of it are largely dependent on software engineering modelling languages. The development process of a KBS is similar to that used in any general system development; stages such as: requirements

gathering, system analysis, system design, system development and implementation are common activities. The stages in KBS development are: business modelling, conceptual modelling, knowledge acquisition, knowledge system design and KBS implementation. Most of the modelling techniques adopt a mix of notations derived from different modelling languages such as: UML, IDEF, SADT, OMT, Multi-perspective Modelling and others. The object-oriented paradigm has influenced systems development activities in software engineering and this trend has also been reflected in knowledge engineering methodologies such as: CommonKADS [2], MOKA [6] and KBS developments in general as described by Felfernig *et al.* [7].

As there is no standard way of modelling knowledge systems, there is a need to extend the use of standardised software engineering modelling techniques such as UML for knowledge modelling. This promotes the use of a common modelling language, so that the vision of integration, reusability and interoperability among enterprise systems will be achieved.

III. UNIFIED MODELING LANGUAGE

The Unified Modeling Language (UML) together with the Object Constraint Language (OCL) is the *de-facto* standard for object modelling in software engineering as defined by the Object Management Group (OMG). The UML is a general-purpose modelling language that may be used in a wide spectrum of different application domains. The OMG [8] has defined two mechanisms for extending UML: profiles and meta-model extensions.

Profiles are sometimes referred to as the "lightweight" extension mechanism of UML [9]. A profile contains a predefined set of Stereotypes, TaggedValues, Constraints, and notation icons that collectively specialize and tailor the UML for a specific domain or process. The main construct in this 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 defined by the meta-model that describes it. However, the usage of stereotypes is restricted, as changes in the semantics, structure, and the introduction of new concepts to the meta-model are not permitted [10].

The "heavyweight" extension mechanism for UML (known as the meta-model extension) is defined through the Meta-Object Facility (MOF) specification [11] which involves the process of defining a new meta-model. Using this extension, new meta-classes and meta-constructors can be added to the UML meta-model.

The "heavyweight" extension is a more flexible approach as new concepts may be represented at the meta-model level; while the "lightweight" extensions are not able to extend the UML meta-model, since they must comply with the standard semantics of the UML meta-model[11]. However, this extension is much more difficult to use compared with stereotypes. It is unfortunate that both extensions are known as profiles.

The work presented in this paper incorporates both the lightweight and heavyweight extension mechanisms of UML using the XMF approach when designing the profile. This is an alternative approach as it allows both mechanisms to be utilised; this contrasts with standard UML that restricts this type of combination. A brief introduction of XMF is given in Section IV.

IV. PROFILE DESIGN – THE XMF APPROACH

The XMF (eXecutable Meta-modelling Language) is an object-oriented meta-modelling language, and is an extension to existing standards for meta-models such as MOF, OCL and QVT, which are also defined by OMG. XMF exploits the features of these standards and adds a new dimension that allows them to be executable using an associated XMT software tool. The most comprehensive use of these standards are seen in the UML in which its' meta-models are described using MOF. Details of XMF can be found in [12].

The XMF approach to creating a profile can be divided into three steps: the derivation of an abstract syntax model, a description of the semantics, and a presentation of the profile's concrete syntax.

A. Abstract Syntax

The abstract syntax model describes the concepts in the profile and their associations. It defines the rules that determine its validity. The processes involved in creating the abstract syntax model are:

- Identifying the concepts including the related rules. Reusing an existing BNF definition of the profile domain is an alternative at this stage.
- Modelling concepts – this involves the process of creating an abstract syntax model using the identified concepts.
- Defining the well-formed-ness rules of the profile in OCL – this will help in ruling out illegal models.
- Defining the operation and the queries related to the profile.
- Validating and testing the profile using an object diagram and relevant tools.

B. Semantics

The semantics describe the meanings of concepts within the profile in terms of behaviour, static properties or how it may be translated into another language. The semantics are a core part of the profile's meta-model and replace formal (mathematical) methods that are often difficult to comprehend by the majority of users and with which it would be difficult to describe the interrelationships within the meta-model. In XMF there are four types:

- Translational – concepts in one language are translated into the concepts of another language that has precise semantics.
- Denotational – modelling the mapping to semantic domain concepts
- Operational – modelling the operational behaviour of language concepts.
- Extensional – extending the semantics of existing language concepts.

C. Concrete Syntax

The concrete syntax is a means of presenting the abstract syntax to end users of the profile, using either textual or diagrammatic forms.

- The textual form of the profile is modelled using Extended Backus-Naur Form (EBNF).
- The diagrammatic form, which involves synchronised mapping between the modelling elements and the diagram elements (boxes, lines and shapes). This is a new technique introduced into the meta-model by XMF.

V. KNOWLEDGE MODELLING PROFILE

The scope of the profile is adapted from [13]. The aim of the XMF Knowledge Modelling Profile is to define a language for designing, visualizing, specifying, analysing, constructing and documenting the artefacts of knowledge-based systems. The profile is based on the XMF specifications and is defined using the meta-class sub-classing approach of the XMF core meta-model, XCore. The knowledge modelling profile is designed using the XMF approach described earlier in Section IV. This paper only concentrates on the creation of the abstract syntax model of the profile. It excludes the processes of defining operations, queries and tool validation for the profile, as these discussions are more appropriate when executing the models and this is not the primary motivation of this paper.

A. Abstract Syntax – Concept Identification

The main thrust in this section refers to the CommonKADS methodology for KBS development [2] and related discussion in [14]. Tasks are the main categorisation of action that needs to be performed by the KBS; typically this refers to the "what we want the system to do". Currently, the development of profiles for modelling knowledge concentrates only on certain task types such as product design in MOKA [6] and product configuration design [7]. As there has been no specific study into creating a generic profile that can be used for different task types, this is the focus of work now underway at the University of York. The following important knowledge modelling concepts have been identified from the literature [2], [14] and are itemised in Table 1.

TABLE I. KNOWLEDGE MODELLING CONCEPTS

Modelling Concepts	**Descriptions**
Concept (Class)	Class that represents the category of things related to knowledge elements
Inference	Describes the lowest level of functional decomposition on carrying out primitive reasoning steps
Transfer Function	Transfers information between the reasoning agent and external entities (system, user)
Task	Defines the reasoning function and invokes the corresponding task method
Task Method	Describes the realisation of the task through sub-function decomposition which includes the invocation of operations on dynamic role, inferences and transfer functions
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	Categorisation and specification of knowledge
Rule	Expressions about an attribute value of a concept
Knowledge Base	Collection of data stores that contains instances of domain knowledge types

B. Abstract Syntax- Syntax Model

The abstract syntax of the knowledge modelling language has been derived using the modelling concepts shown in Table 1. The CommonKADS language has been adopted for specifying knowledge models that are defined in the BNF notation [2]. That BNF description has been translated into a UML model. In its current form it is a model of the abstract syntax of a knowledge modelling language, becoming a complete model of the language: a meta-model. Due to the size, and repetitive nature of the concepts described using BNF, and the complexity of the model, it has been condensed to show only the important features of modelling knowledge concepts.

Shown in Fig. 1 is the knowledge modelling profile that is composed of four main packages based on their role and their interrelationships. It consists of the Domain Concept package, Inference package, Knowledge Base, and Rule Type package.

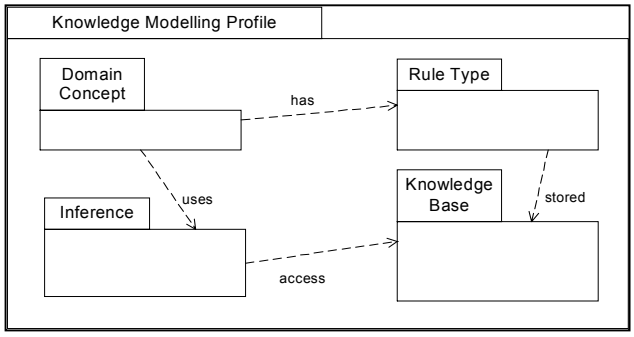

Figure 1. Knowledge Modelling Profile Core Package

The Domain Concept package within the Knowledge Modelling package describes the concept constructs of the profile that are related to knowledge elements. This package is shown in Fig.2.

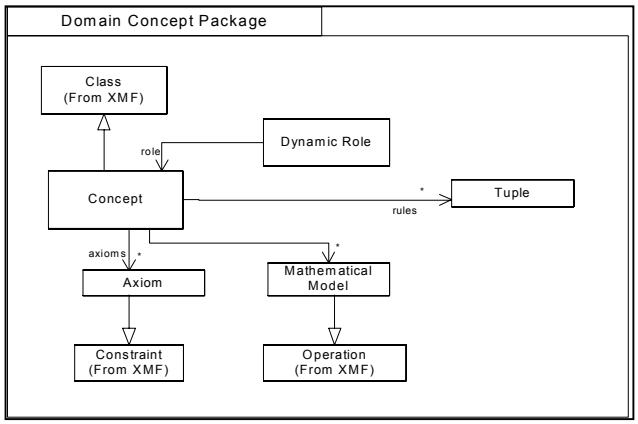

Figure 2. Domain Concept Package

The Inference package of the profile describes the inference, inference method, task, task method, transfer function and both the static and dynamic knowledge roles. The inference package plays a pivot role in designing KBS as it defines the inference structure of the system, the type of knowledge used in the reasoning process and the task associated with the execution of the inference. An important point to note here is that the KBS is designed independently of the target implementation platform and inference engines, overcoming the difficulties of reusing implementation specific designs. This package is shown in Fig.3.

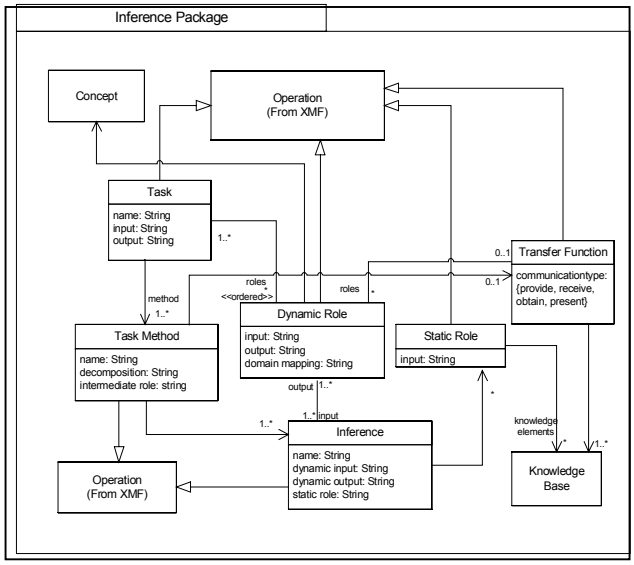

Figure 3. Inference Package

The Knowledge Base package of the profile describes the modelling of a knowledge base that represents instances of knowledge elements (instances of rule type) of the domain concepts. These instances are important as they contain the actual knowledge on which the KBS reasoning process is based. Knowledge elements within the knowledge base are accessed by an inference through static role. This package is shown in Figure 4.

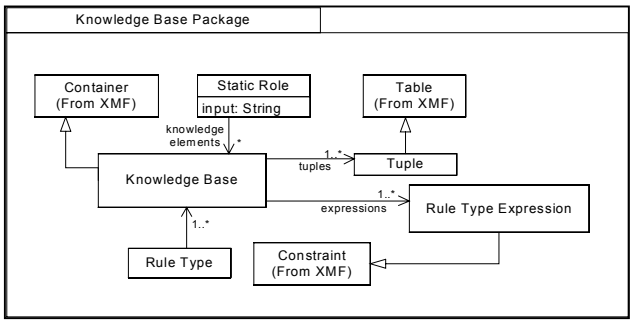

Figure 4. Knowledge Base Package

The Rule Type package (shown in Fig. 5) within the profile describes the modelling of rules. There are three types of rules: constraint rule, implication rule and decision table. Decision table is an addition to rule type that is introduced here, as certain rules are in the form of a decision table. Currently, we are only concentrating on rule-based KBS and Case-Based Reasoning (CBR) is out of the profile scope.

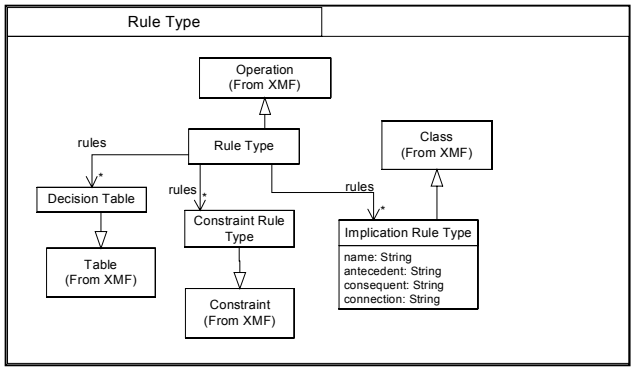

Figure 5. Rule Type Package

C. *Abstract Syntax- Model Extension*

The knowledge modelling profile concept extends the existing meta-models of XMF by defining the profile's abstract syntax. There are five places where the profile can be viewed as an extension to XMF and these are: Class, Operations, Container, Table and Constraints from the Core XMF meta-model.

The knowledge modelling class concept is viewed as a special class that is a subclass of the XMOF Class. This enables the concept to inherit all the features of a class and allows it to define additional constraints such as "concepts do not have any operations or methods". The implication rule type is also another example of this.

Constraint class is another area where we subclass XMF meta-model to incorporate profile concepts such as axioms, rule type expression and constraint rule type. All these concepts need the ability to express constraints and this class allows for constraint expressions. For example, axioms are often used to define specification of a (mathematical) relationship that is defined to be true, and the constraint class is a natural choice as it allows constraint expressions of axioms.

The inference package of the profile (which has the task, task method, inference, dynamic role, static role, and the transfer function concepts) can be viewed as a subclass of an XMF Operations class. The same is true for the mathematical model in the domain concept package. The operation class of XMOF allows operations related to objects to be expressed, such as execute inference call from task method, execution of the inference process and accessing the knowledge in the knowledge base through static role.

Knowledge base is viewed as a subclass of the Container class of XMF. It has a 'content' slot that is a table. 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 table class of XMF is extended to incorporate the profile's concepts of tuple and decision table (in which is stored rule type instances). The table class is a new feature in the meta-modelling that was introduced by XMF

D. Abstract Syntax- Well-formed-ness Rules

The following well-formed-ness rules are defined for each of the modelling concepts that have been introduced in the profile.

TABLE II. WELL-FORMED-NESS RULES

Class	Well-formed-ness rule descriptions
Concept	Concept doesn't have any operations/methods. Concept must exist as a representation of the object that has knowledge associated with it.
Axiom	Axiom values are defined by the concept's attribute.
Task	Task must exist. Task must have unique name.
Task Method	Task method must exist. Task method must have unique name. Task method may define additional task roles to store temporary reasoning results. Task method decomposition can either be another task, an inference, or a transfer function.
Inference	Inference must exist. Inference must have unique name. Inference must have dynamic input and output. Inference may not have static input.
Transfer Function	Transfer function type may only be: obtain, receive, present and provide.
Dynamic role	Dynamic role must exist Dynamic role must have an input and output.
Static role	Static role must exist Static role must have an input and output.
Rule Type	Rule type must exist Any one of the rule types must exist: constraint, implication and decision table.
Constraint rule	Constraint rule can either be: single constraint, multiple constraint, or grouped constraint.
Implication rule	Implication rule must have antecedent and consequent. Antecedent can be more than one.
Decision table	Table is a two dimensional table.
Knowledge Base	Knowledge base must exist. Knowledge base must contain at least one tuple. Knowledge base must contain instances of at least one rule type. Only static role can access the knowledge base.

An example of one of the rules written in XOCL (an executable subset of OCL) 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
```

E. Example of Housing Application Assessment

A brief description of the housing application case study given in [2] follows. Rental residences are allocated to potential applicants based on four types of eligibility criteria. First, people have to apply for the right residence category. Second, the size of the household of the applicant needs to be consistent with the requirements on minimum and maximum habitation in a certain residence. The third criterion is that there should be a match between the rent of the residence and the income of the applicant. Finally, there can be specific conditions that hold for one particular residence.

Fig. 6 shows part of the knowledge modelling profile used to represent the assessment in the housing application case study. This example only concentrates on showing the abstracting process of the residence application. The purpose of this abstraction process is to provide useful categories of cases that need to be distinguished for assessment purposes. Here the assessment task will abstract all cases into two groups, thus allowing a relatively large set of cases to be categorised. The example here is translating the original knowledge model (described in CommonKADS language) into an XMF class diagram based on the abstract syntax model shown earlier in this section.

The profile packages used here are the domain concept, inference, knowledge base and rule type. There are two concepts that represent the residence and applicant, and between these concepts an association class residence-application. For reasons of space, only one particular attribute related to each concept and its associated axioms are shown in the diagram.

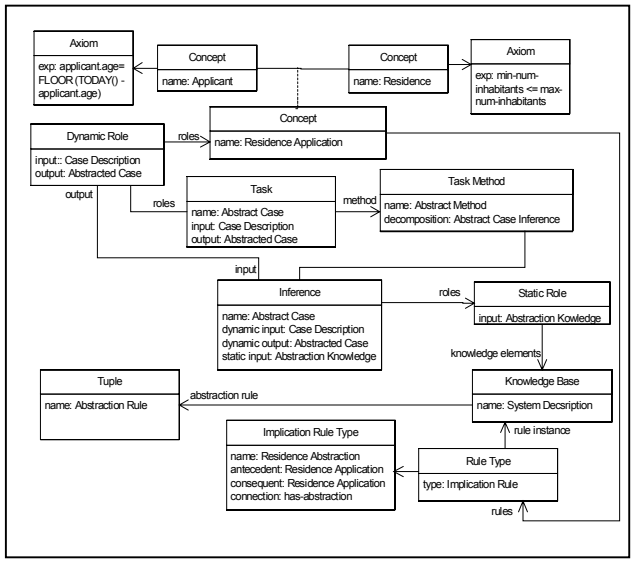

Figure 6. Housing Application Assessment

The inference package will execute the task of abstracting the cases by adding case abstraction to the case data. The task will be realised by the task method "abstract case". The task method can be decomposed into other tasks or inferences. In this example it is decomposed into the inference "abstract". The input for this task is "case descriptions" and the output will be "abstracted case". The inference will carry out the reasoning process of abstracting the cases. The knowledge role here will have the same input and output as the task, but with inferences, they are referred to as dynamic input/output. The reasoning process will use the "abstraction knowledge" which is a static knowledge role. This knowledge is accessed from the knowledge base.

In knowledge modelling, all the processes and actions carried out by the system is specified in detail to help the KBS

developer understand the working mechanisms of the system being designed. An important feature here is that it has been explicitly stated the knowledge base used by the inference and the rule type associated with it. The existence of knowledge base is a typical characteristic of knowledge modelling. The knowledge modelling profile is designed to allow this type of specification and it is exploited in the housing application assessment example.

VI. Conclusion and Future work

Managing knowledge through knowledge-based systems is an important part of an enterprise's knowledge management initiative. Systems of this sort have evolved from being stand-alone machines to being part of the enterprise's group of systems. The process of constructing knowledge based systems is similar to that required by other software systems, but conceptual modelling plays an important role in the development process. Software engineering has adopted UML as a standard for modelling, but the field of knowledge engineering is still searching for the right technique. UML can be adopted for knowledge modelling by exploiting the profile extension mechanism defined by OMG.

This paper has described the process of creating such an extension by basing the design of the knowledge modelling profile on that of the XMF framework. This is a novel approach in profile design as the XMF approach is an extension to existing standards for meta-modelling such as MOF, OCL and QVT, which are defined by OMG. This approach is similar to UML, which has it models defined by MOF, and XCore, which defines XMF is an extension to MOF. The creation of a profile is important as it allows knowledge based systems to be designed using an object-oriented approach.

The knowledge modelling profile has defined concepts which are used to develop the abstract syntax model of the profile. This allows the capture of modelling elements associated with the knowledge engineering domain in respect to KBS design and the relationship between these concepts. The profile's well-formed-ness rules have been identified and allowing for additional constraints, related to the concepts, to be defined. An example demonstrates the ability of this profile to model a knowledge based system.

Developing a profile is not an easy task and involves many steps as demonstrated in Section IV of this paper. The future work in this area involves the specification of the profile's semantics and construction of the concrete syntax model. Both activities involve the use of the XMT tool, which is in its final stage of development. The profile will be validated using this tool and the models executed wherever possible.

The profile's ability to model the requirements of a knowledge based system has only be tested on a case study adapted from one available in the literature. Testing the profile in a number of real-world situations would be beneficial and would identify any limitations and assist in the refinement of the profile.

Acknowledgment

The authors gratefully acknowledge the provision of a fellowship from Universiti Utara Malaysia that has enabled this research to take place, and are grateful to Xactium for early access to XMF. Details of XMF can be found at http://albini.xactium.com.

References

[1] Studer, R., Benjamins, R.V., and Fensel, D.: *Knowledge Engineering: Principles and Methods.* Data & Knowledge Engineering, 1998. **25**: p. 161-197.

[2] Schreiber, G., Akkermans, H., Anjewierden, A., de Hoog, R., Shadbolt, N., de Velde, W.V. and Wielinga, B.: *Knowledge Engineering and Management: The CommonKADS Methodology*. 1999, Massachusetts: MIT Press.

[3] Angele, J., Fensel, D., Landes, D., Studer, R.: *Developing Knowledge-Based Systems with MIKE.* J of Automated Software Engineering, 1998. **5**(4): p. 389-418.

[4] Grosso, W.E., Eriksson, H., Fergerson, R.W., Gennari, S., Tu, S., and Musen, M.A.: *Knowledge Modelling at the Millennium (The Design and Evolution of Protege 2000)*. 1999, Stanford Medical Institute.

[5] Speel, P., Schreiber, A. Th., van Joolingen, W., and Beijer, G.: *Conceptual Models for Knowledge-Based Systems*, in *Encyclopedia of Computer Science and Technology*. 2001, Marcel Dekker Inc, New York.

[6] Stokes, M., Managing Engineering Knowledge: MOKA - Methodology for Knowledge Based Engineering Applications. 2001, London, UK: Professional Engineering and Publishing Limited.

[7] Felfernig, A., Friedrich, G.E., Jannach, D.: Generating product configuration knowledge bases from precise domain extended UML models. in 12 th International Conference on Software Engineering and Knowledge Engineering (SEKE'00). 2000. Chicago, USA.

[8] OMG: Unified Modeling Language specification (version 1.4). 2001.

[9] OMG: Requirements for UML Profile. 1999.

[10] Perez-Martinez, J.E.: Heavyweight extensions to the UML meta-model to describe the C3 architectural style. ACM SIGSOFT Notes, 2003. **28**(3).

[11] OMG: MOF Specification version 1.4. 2002.

[12] Clark, T., Evans, A., Sammut, P., Willians, J.: Applied Metamodelling : A Foundation for Language Driven Development (draft ver 0.1): To be published. 2004. Accessible at http://albini.xactium.com/

[13] OMG: UML 2.0 Testing Profile specification. 2003.

[14] Abdullah, M.S.: *Extending UML Using Profile for Knowledge-Based Systems Modelling*. 2004, Thesis Proposal, Computer Science, University of York: York.