Developing a UML Profile for Modelling Knowledge-Based Systems

Mohd Syazwan Abdullah, Andy Evans, Ian Benest, Chris Kimble

Department of Computer Science, University of York,
Heslington, YO10 5DD, United Kingdom
{syazwan, andye, kimble, idb } @cs.york.ac.uk

Abstract. Knowledge engineers have favoured a diagrammatic approach for developing knowledge-based systems by adopting those used in software engineering. However, these modelling techniques tend to be used in an *ad hoc* way and are highly dependent on the modelling experience of the engineers involved. This paper focuses on the use of the Unified Modeling Language (UML) Profiles for knowledge modelling. It identifies the short-comings of current approaches in adopting UML and discusses the need to have an extension to UML through the profile mechanism. A work-in-progress on creating such a profile is also presented.

1 Introduction

The use and management of knowledge in enterprises has become a commercial necessity for many enterprises, in order that they manage their corporate intellectual assets and gain competitive advantage. Most knowledge resides in human memories and managing it is seen as a human-oriented process rather than a technology-based solution. Nevertheless, technology can be utilised as a knowledge management enabler with automated tools, including the internet and groupware systems. One of the prominent tools in managing knowledge is knowledge-based systems (KBS).

Knowledge-based systems can be deployed as the technological means for capturing and managing both explicit and tacit knowledge as part of an organisation's knowledge management initiative. But, before these can be built, the knowledge that pervades the organisation must be identified and modelled using appropriate acquisition, representation and modelling techniques.

This paper is organised as follows: Section II describes KBS and the field of knowledge engineering. Section III gives an overview of the rôle of knowledge modelling and the techniques that are currently used. Section IV explains the need to have an extension to UML for modelling knowledge, while Section V describes what is a UML profile. Section VI presents the initial knowledge modelling profile constructed using identified modelling concepts, while Section VII concludes and indicates the direction for future work.

2 Knowledge-Based Engineering and Knowledge Engineering

A KBS is a software application with an explicit, declarative description of knowledge for a certain application [1]. There is no single dividing line that differentiates a KBS and an information/software system as almost all contain knowledge elements in them [2]. An information system is a set of interrelated components that together collects, processes, stores, analyses, and disseminates data and information in an organization. In contrast, a KBS has knowledge represented in an explicit form, and hence the increased importance of knowledge modelling [2] compared with that required of an information system.

The development process of a KBS is similar to 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 [1].

A KBS is developed using knowledge engineering (KE) techniques [3]. These are similar to software engineering (SE) techniques, but have an emphasis on knowledge rather than data or information processing and they inherently advocate an engineering approach to the process of developing a KBS. The central theme in this approach is the conceptual modelling of the system in the analysis and design stages of the development process. Many knowledge engineering (KE) methodologies have been developed with an emphasis on the use of models, for example CommandKADS [2], MIKE [4], Protégé [5], and KARL [4].

Traditional KE techniques were widely used to construct expert systems – systems built from the knowledge of one or more experts – essentially, a process of knowledge transfer [3]. This is the development process of the first generation of expert systems, in which the knowledge of the expert is directly transferred into the knowledge base in the form of rules. The disadvantage of this approach is that the knowledge of the expert is captured in the form of hard codes within the system with little understanding of how they are linked or connected with each other [2]. This creates a new problem if the knowledge base is to be updated as changes require substantial effort in reconstituting the coded rules in order to implement the needed changes.

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]. The shift towards the modelling approach has also enabled knowledge to be re-used in different areas of the same domain [3]. In the past, most knowledge systems had to be developed from scratch every time a new system was needed, and it could not interact with other systems in the organization. The paradigm shift towards a modelling strategy has resulted in reducing development costs [2].

3.0 Knowledge Modelling

"*A model is a simplification of reality*" [6]. Real systems are large entities consisting of interrelated components working together in a complex manner. Models are used both to build descriptions of the problem domain in software and to define the systems development process [7]. Models help people to appreciate and understand such complexity by enabling them to look at each particular area of the system in turn. The value of a model in the context of systems development is dependent on the effects it has on the systems being produced. Models capture the essential features of real systems by partitioning them into components that are easy to understand and to manipulate. It is very difficult for the human mind to be able to capture all the features of a system as a mental model and then convey them in either written or oral form. The mind often works better with a visual representation. Models are very much associated with the domain they represent. That domain will define their practicing communities, modelling languages and their associated tools. Each domain will have their own techniques for representing concepts associated to that domain. To model the system, there is a need for a language to express the description of the system [8]. Modelling languages are also used in the process of modelling knowledge when developing knowledge-based systems.

Knowledge modelling is used in knowledge acquisition activities as a way of structuring projects, acquiring and validating knowledge and storing knowledge for future use [9]. Knowledge models are structured representations of knowledge. They use symbols to represent pieces of knowledge and their relationships. Knowledge models are as follows: (1) symbolic character-based languages – logic; (2) diagrammatic representations – networks and ladders; (3) tabular representations – matrices and frames and (4) structured text – hypertext. Most models are constructed from knowledge objects such as concepts, instances, processes (tasks, activities), attributes and values, rules and relations.

Knowledge representation is one of the fundamental topics in the area of artificial intelligence (which investigates representation techniques, tools and languages). Knowledge about the domain and the implementation independent reasoning-process of the KBS however is usually addressed through the use of ontologies and problem-solving methods. There are five prominent representation techniques widely used in developing KBSs; they are attribute-value pairs, object-attribute-value triplets, semantic networks, frames and logic.

By analysing the knowledge objects and representation techniques described earlier in this section, it will be noticed that they have similar concepts to those adopted for object-oriented modelling. Examples of these concepts are objects, attributes, class, subclass, relationship, instances and others. Though these concepts have different meanings in different techniques, in most cases they refer to a similar thing. This paves the way to consider using object-oriented techniques as the standard means of representing them.

3.1 Ontology and Problem-Solving Methods

Ontologies and Problem-Solving Methods (PSMs) enable the construction of KBSs through reusable components across domains and tasks [8]. Systems developers in the KE community are currently trying to adopt component-based development by incorporating ontologies and PSMs in order to deploy KBSs faster.

Ontologies are used to represent domain knowledge in knowledge-based programs. This is achieved using formal declarative representations of the domain knowledge; that is sets of objects and their describable relationships [11]. In the context of knowledge modelling, ontology defines the content-specific knowledge representation elements such as domain-dependent classes, relations, functions and object constants [10]. Researchers in the area of conceptual modelling and knowledge modelling have started to realise the importance of ontology in developing domain models since the underlying principle of modelling is to achieve agreed representations in a unified manner for the domains in which they are investigating. The works of [10], [11] and [12] demonstrate such efforts on the usage of ontologies.

PSMs describe the reasoning-process (generic inference patterns) at an abstract level independent of the representation formalism (e.g. rules, frames etc) [5], [10]. PSMs have influenced the leading knowledge-engineering frameworks such as Task Structures, Rôle-Limiting Methods, CommonKADS, Protégé, MIKE, Components of Expertise, EXCEPT, GDM and VITAL [10]. Most of these frameworks suggest that a PSM: decomposes the whole reasoning task into elementary inferences that are easy to understand, defines the types of knowledge that will be used by the inference steps to be completed, and defines the control mechanisms and flow of knowledge among the inferences.

3.2 Knowledge Modelling Techniques

The importance of knowledge modelling in developing KBSs has been discussed in [2]. They 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. Modelling contributes to the understanding of the source of knowledge, the inputs and outputs, the flow of knowledge and the identification of other variables such as the impact that management action has on the organizational knowledge. Using conceptual modelling, systems development can be faster and more efficient through the re-use of existing models for different areas of the same domain. Therefore, understanding and selecting the modelling technique that is appropriate for different domains of knowledge will ensure the success of the KBS being designed.

Amongst the many techniques used to model knowledge, the most common are CommonKADS, Protégé 2000, the Unified Modeling Language (UML), and Multi-perspective modelling.

CommonKADS has become the *de facto* standard for knowledge modelling and is used extensively in European research projects. It supports structured KE techniques, provides tools for corporate knowledge management and includes methods that perform a detailed analysis of knowledge intensive tasks and processes. A suite of models is at the core of the CommonKADS methodology [2]. The suite supports the modelling of the organization, the tasks that are performed, the agents that are responsible for carrying out the tasks, the knowledge itself, the means by which that knowledge is communicated, and the design of the knowledge management system. CommonKADS incorporates an object-oriented development process and uses UML notations such as class diagrams, use-case diagrams, activity diagrams and state diagrams. CommonKADS also has its own graphical notations for task decomposition, inference structures and domain schema generation [2].

It has become a trend for system developers and researchers in KE to adopt object oriented modelling in developing conceptual models for knowledge systems [13] [14] [15]. A careful analysis of the literature shows that they have all been influenced by CommonKADS – an approach that is highly favoured, since it encourages the use of object-oriented development and the notations from UML.

Protégé was developed for domain specific applications [5] at Stanford Medical Informatics. Protégé 2000 is defined as "*an extensible, platform-independent environment for creating and editing ontologies and knowledge bases*" [16]. The Protégé 2000 knowledge modelling environment is a frame-based ontology editing tool with knowledge acquisition tools that are widely used for domain modelling.

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 covers a wide spectrum of different application domains. UML is incorporated in other mainstream techniques such as CommonKADS and Multi-perspective modelling for knowledge modelling purposes. Multi-perspective modelling enables a number of techniques to be used together, each technique being the most appropriate for modelling that particular aspect of knowledge [17]. It has its roots in software engineering (multiple-view technique).

3.3 Current Trends

Although KBSs are developed using knowledge engineering techniques, the modelling aspects of it are largely dependent on software engineering modelling languages. Most of the modelling techniques adopted a mix of notations derived from different modelling languages. 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], Methodology and tools Oriented to Knowledge-based engineering Applications (MOKA) projects [15] and KBS developments in general as shown in the works of [13], [18] and [19]. However, the main adopters of UML for knowledge modelling are CommonKADS

[2] and MOKA [15]. The MOKA Modelling Language (MML) is an extension of UML that represents engineering product design knowledge at a user level for deployment in knowledge-based engineering applications. It provides default meta-models for the product and design process so as to manage engineering knowledge. However, it is an informal extension to UML and does not fulfill the OMG's requirements for an extension mechanism; these are presented in the section 5.

Object oriented methods are gaining in popularity because of their expressiveness, flexibility and ease of use. One of UML's important features is that it is an extensible language brought about by the application of profiles. This makes UML one of the favoured techniques for knowledge modelling, for both the methodological aspect of KBS development and its standardisation. Thus, extensions to UML, can be formally introduced using UML Profiles for knowledge modelling.

4 Need for UML Extension

The major problem with knowledge modelling is that there is no standard technique available to model the knowledge for developing a knowledge based system. Most of the techniques used by the researchers in the field of knowledge engineering are adapted from the software engineering community. The techniques used in knowledge modelling are project based using a mix of notations such as UML, IDEF, SADT, OMT, Multi-perspective Modelling and so on. Examples mentioned earlier are the CommonKADS methodology and Multi-perspective Modelling.

Another important factor to consider is that most system analysis and design courses these days are teaching object-oriented modelling techniques as a tool for systems modelling and development. The main influence is the growing importance of object-oriented programming languages like Java in systems development. Due to the formal training received and the adoption of object-oriented programming by this generation of system analyst, most will have the knowledge of UML and use them for modelling purposes.

In addition to this, enterprise systems these days are an integration of various systems built on different platforms with the ability to communicate with each other. Most of these systems especially the new ones are built on platforms that support object-oriented languages, model driven architectures, object-based modelling etc. Knowledge-based systems are no longer stand-alone systems, but are part of the enterprise group of systems. As there is no standard way of modelling knowledge systems using knowledge engineering techniques, there is a need to extend those that have been standardised in software engineering. This promotes the use of a common modelling language, so that the vision of integration, reusability and interoperability within an enterprise's system will be achieved. It is proposed to model knowledge using an extension to UML.

UML is widely adopted as the object oriented way for systems development and has been deployed in other domains such as real-time systems, hypermedia design, embedded systems and ontology modelling. There are arguments that UML semantics are not well defined [20][21] compared to formal methods and these are being addressed by the OMG in developing UML version 2.0 that will have enhanced meta-model concepts and unambiguous semantics. Developing UML Profiles for knowledge modelling will enable KBSs developers to use UML in a formal and systematic manner. This can be achieved through the means of developing UML profiles with precisely defined notations, semantics and syntax which enables this extension to be formally integrated into the existing profiles of UML, and adheres to the profiles requirements proposed by OMG [22].

The UML is a general-purpose modelling language that covers a wide range of different application domains. While this feature might be adequate for modelling in a broader area, some domain-specific concepts and techniques need a more specialised refinement to the existing construct of the language [22]. This is achievable through the usage of the extension mechanism provided by UML known as profiles.

5 Profile Extension Mechanism

The OMG [23] has defined two extension mechanisms for extending the UML: profiles and metamodel extensions. Profiles are sometimes referred to as the "lightweight" extension mechanism of UML [22]. It 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 the profile is the stereotype that is purely an extension mechanism. In the model, it is marked as <<stereotypes>> and has the same structure (attributes, associations, operations) defined by the metamodel that describes it. However, the usage of stereotypes is restricted as changes in the semantic, structure, and the introduction of new elements to the metamodel is not permitted [24]. The "heavyweight" extensions mechanism to UML known as the metamodel extension is defined through the Meta-Object Facility (MOF) specification [25] which involves the process of defining a new metamodel. Using this extension, new metaclasses and metaconstructors can be added to the UML metamodel. This extension is a more flexible approach as new concepts may be represented at the metamodel level. The difference between the profile and metamodel extensions comes from the restrictions on profiles in extending the UML metamodel [25]. These restrictions impose that profile based extensions must comply with the standard semantics of the UML metamodel. However, these restrictions are not applicable to the MOF based extensions, which can define a new metamodel. Nevertheless, both extensions are called profile.

UML Profile for Enterprise Application Integration (EAI), UML Profiles for CORBA, UML Profile for Enterprise Distributed Object Computing (EDOC), UML Testing Profile, and UML Profile for Schedulability, Performance and Time are some of the formal profiles developed by OMG.

6 UML Knowledge Modelling Profile

The scope of the profile described below is adapted from [26]. The aim of the UML Knowledge Modelling Profile is to define a language for designing, visualizing, specifying, analyzing, constructing and documenting the artifacts of knowledge-based systems. It is a knowledge modelling language that can be used with all major object technologies and applied to knowledge-based systems in various application domains and task types. The UML profile is based on the UML 2.0 specifications and is defined by using the metamodelling extension approach of UML. It is being designed with the following principles in mind: (1) UML integration: as a real UML based profile, the knowledge modelling profileis defined based on the metamodel provided in the UML superstructure and follows the the principles of UML profiles as defined in the UML 2.0 and (2) Reuse and minimalist: wherever possible, the knowledge modelling profile makes direct use of the UML concepts and extends them, adding new concepts only where needed.

The discussion in this section mainly refers to the CommonKADS methodology for KBS development [2] and related discussion in [27]. Tasks are the main categorisation of action that need to be performed by the KBS which typically refers to the "what we want the system to do". Each task type will have their own terminology, task methods, inputs, outputs, inference mechanism being used, and the type of knowledge used; this is presented in [2]. Current studies on extending UML to model knowledge only concentrates on certain task types such as product design in MOKA [15] and UML-based product configuration design [13]. There are no specific studies being conducted in creating a generic profile that can be used for different task types; research now underway at York is focusing on this work.

In [28] there are suggestions as to how to construct a modelling language. This involves the creation of an abstract syntax model, identifies and models concepts, specifies well-formed rules and operations, and finally validates and tests the model. The first step in creating the meta-model of the knowledge modelling profile is to build its abstract syntax model. The syntax model is used to describe the concepts of the profile and the relationships between concepts. The concepts will provide a vocabulary and grammar for constructing models in the profile [28]. The following important knowledge modelling concepts have been identified from the literature [2], [27] and are itemised in Table 1.

Table 1. Main Knowledge Modelling Concepts

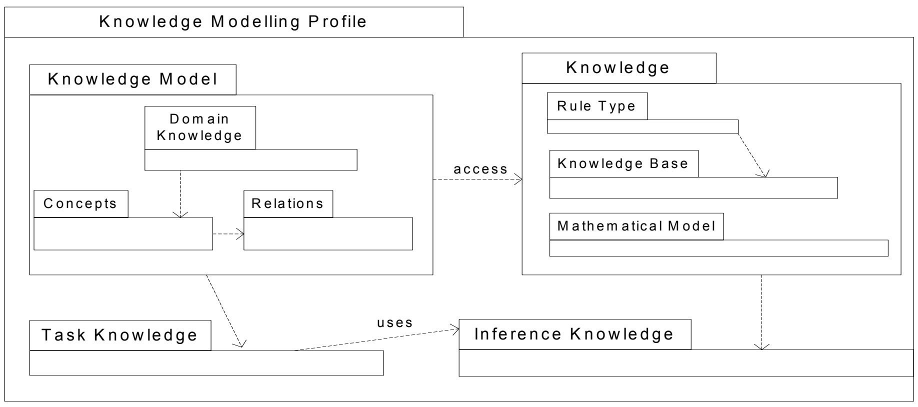
Modelling Concept	**Description**
Concept (class)	Class that represents the category of things
Inference	Describes the lowest level of functional decomposition on carrying out primitive reasoning steps
Inference Method	Method for implementing the inference
Transfer Function	Transfers information between the reasoning agent and external entities (system, user)

Task	Defines the reasoning function
Task Method	Describes the realization of the task through subfunction 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 about an attribute value of a concept
Knowledge Base	Collection of data stores that contains instances of domain knowledge types

The abstract syntax of the knowledge modelling language has been built using these modelling concepts and the CommonKADS language is adopted for specifying knowledge models that are defined in the BNF notation [2]. The BNF notation 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 metamodel. Unless it is viewed as an extension of UML, it is not a profile, but just a plain metamodel. Efforts are currently focused on developing this metamodel further by defining well-formedness rules, syntax and semantics for the language and mapping it to the core UML.

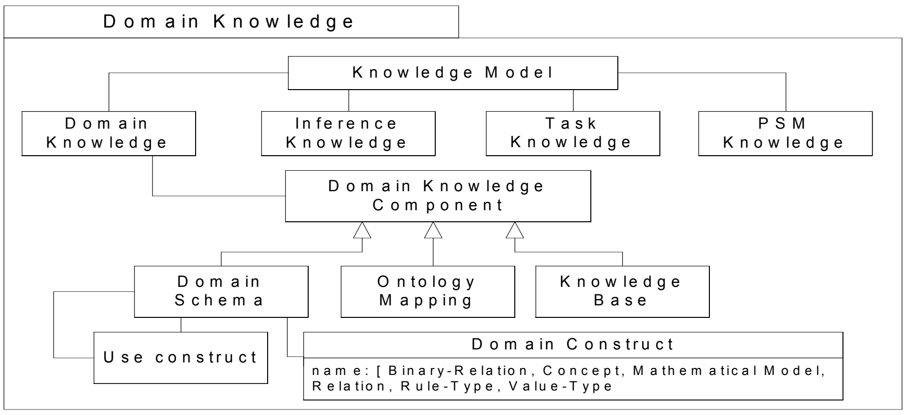
The initial knowledge modelling profile is composed using four main packages based on their rôle and relationship in modelling KBSs. It consists of the Knowledge Model package, Task Knowledge package, Inference Knowledge package and Knowledge package. These packages forms the knowledge modelling language core model and is shown in Figure 1 as the knowledge modelling profile.

Fig. 1. Knowledge Modelling Profile Core Package



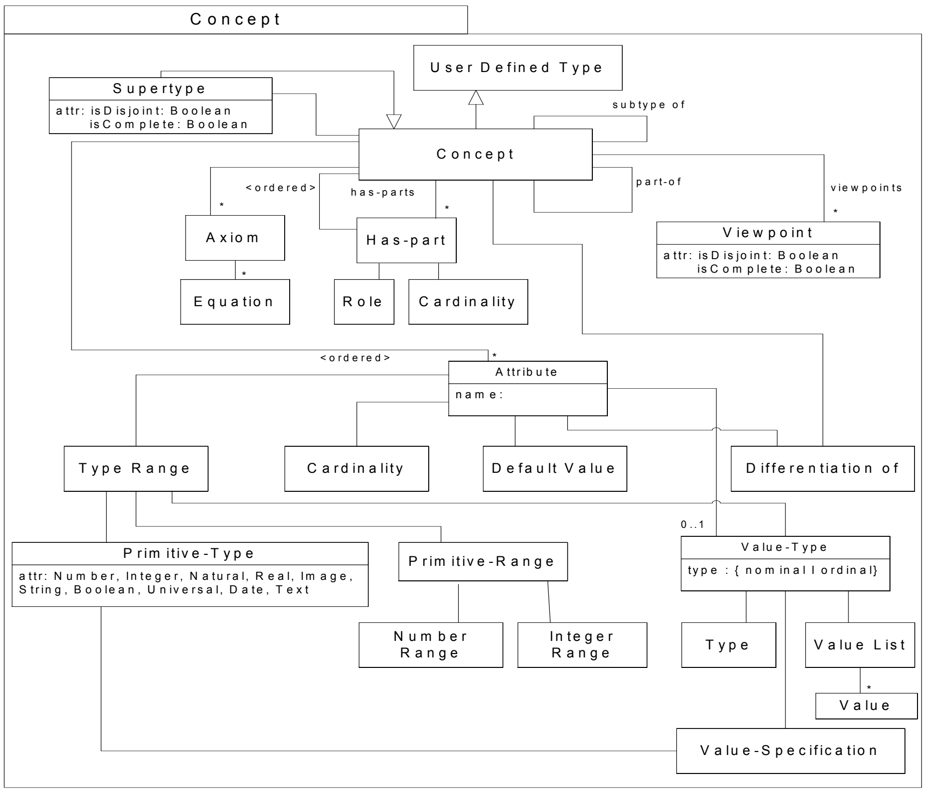
The Domain Knowledge package within the Knowledge Model package describes the main constructs of the profile. This package is shown in Figure 2.

Fig. 2. Domain Knowledge package



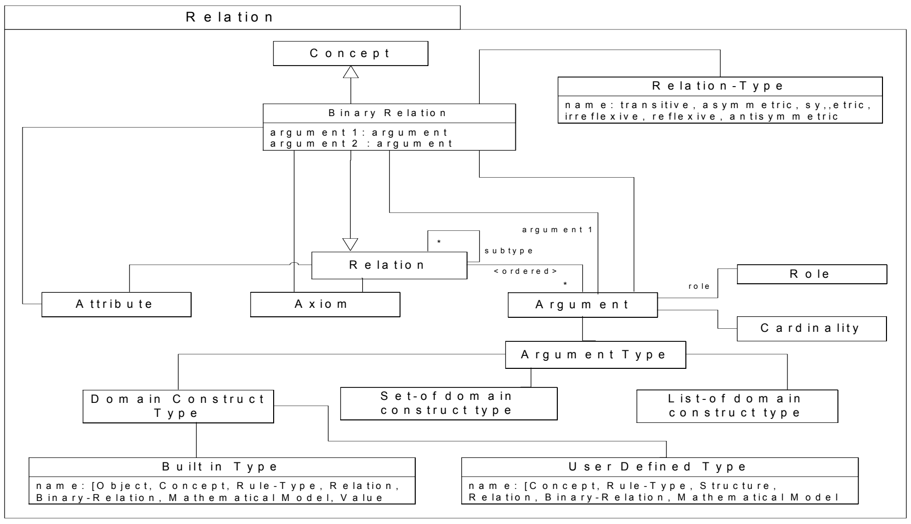
The Concept package within the Knowledge Model package describes the concept of the profile. Concept here represents class. This package is shown in Figure 3.

Fig. 3. Concept Package



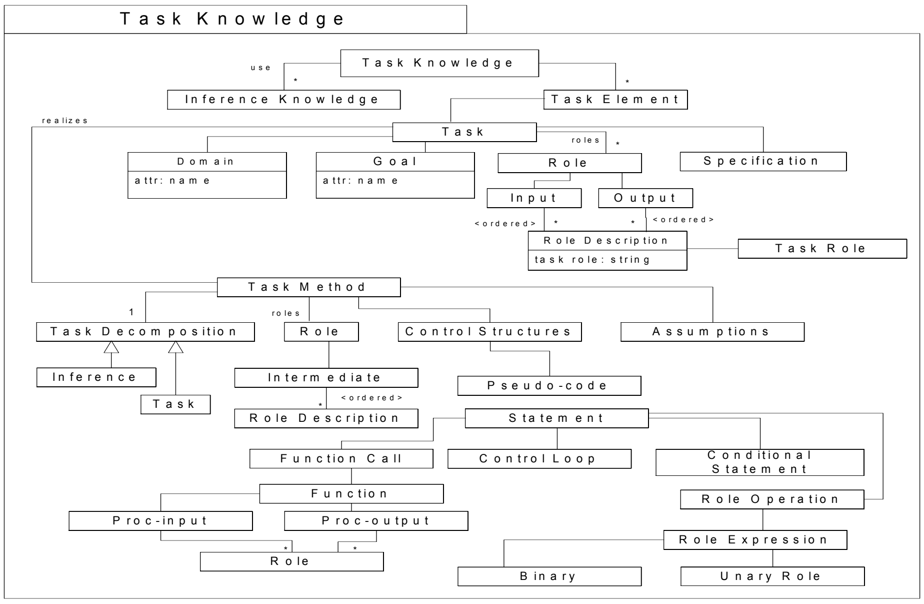
The Relations package within the Knowledge Model package describes the relations in the profile. This package is shown in Figure 4.

Fig. 4. Relation Package


Relation
Concept
Relation-Type
name: transitive, asymmetric, sy,,etric, irreflexive, reflexive, antisymmetric
Binary Relation
argument 1: argument
argument 2 : argument
argument 1
subtype
<ordered>
Relation
role
Role
Attribute
Axiom
Argument
Cardinality
Argument Type
Domain Construct Type
Set-of domain construct type
List-of domain construct type
Built in Type
name: [Object, Concept, Rule-Type, Relation, Binary-Relation, Mathematical Model, Value
User Defined Type
name: [Concept, Rule-Type, Structure, Relation, Binary-Relation, Mathematical Model

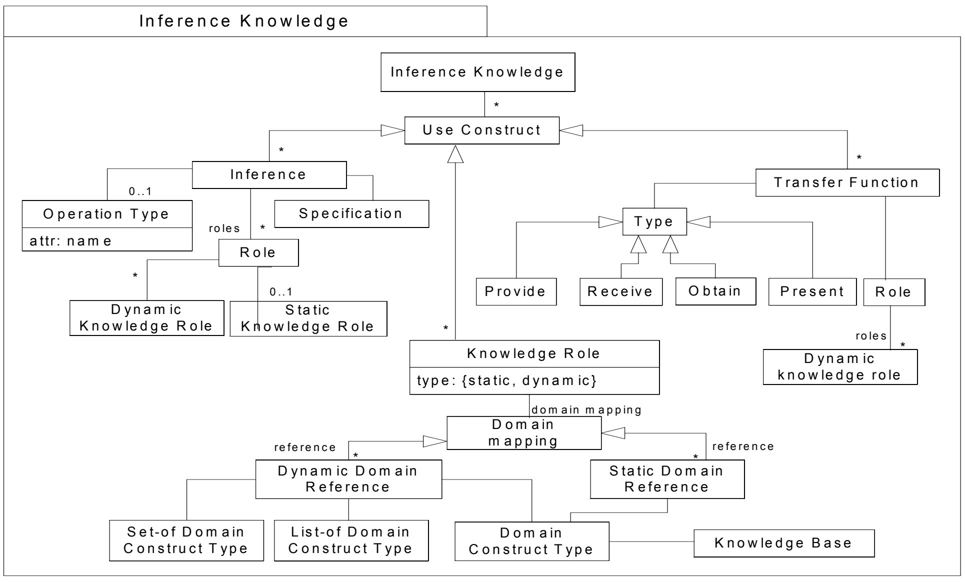
The Task Knowledge package of the profile describes the task and task method in detail. This package is shown in Figure 5.

Fig. 5. Task Knowledge Package



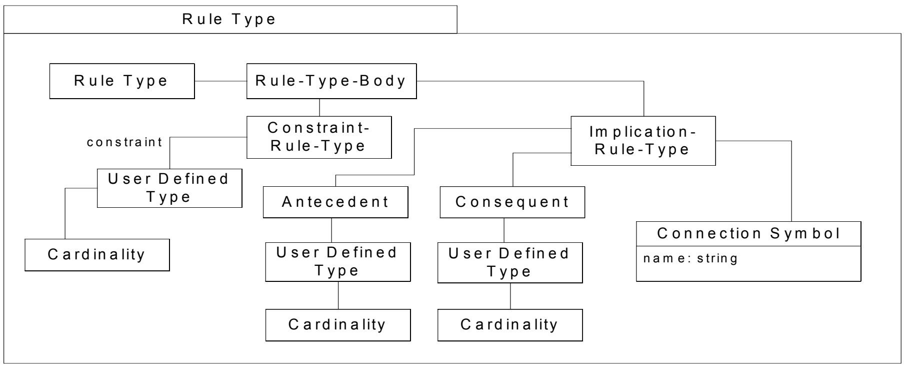
The Inference Knowledge package of the profile describes the inference, knowledge role and transfer function in detail. This package is shown in Figure 6.

Fig. 6. Inference Knowledge Package



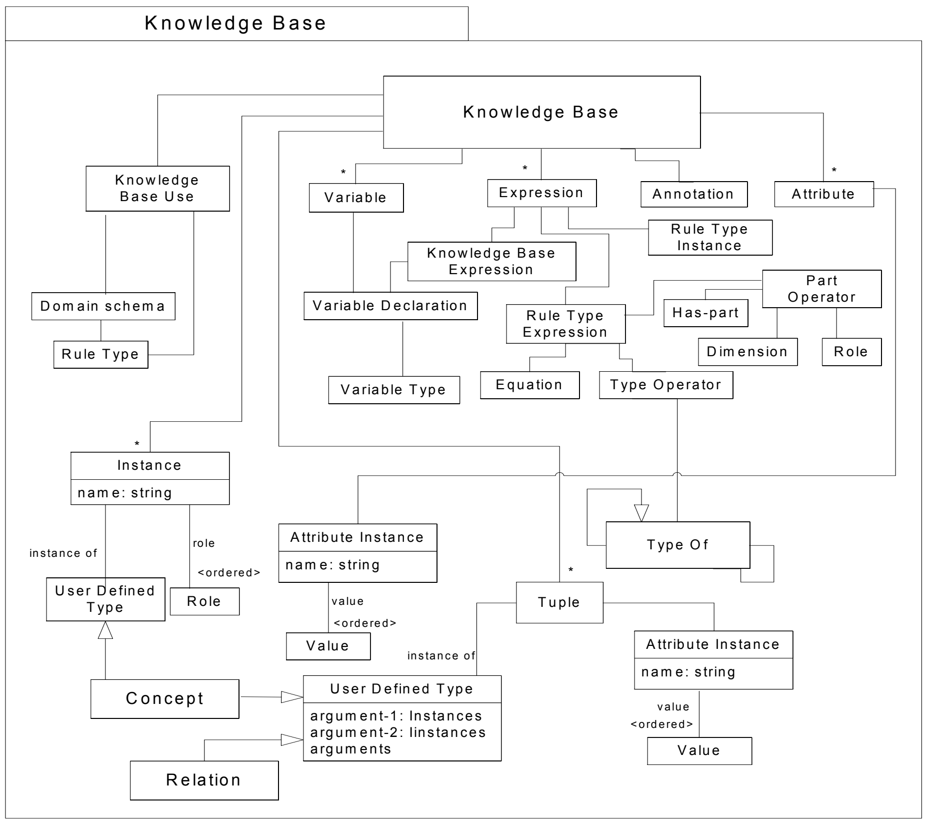
The Knowledge package of the profile is grouped into three packages: the Rule Type package, the Knowledge Base package and the Mathematical Model package. The Rule Type package within the Knowledge package describes the modelling of rules. This package is shown in Figure 7.

Fig. 7. Rule Type Package



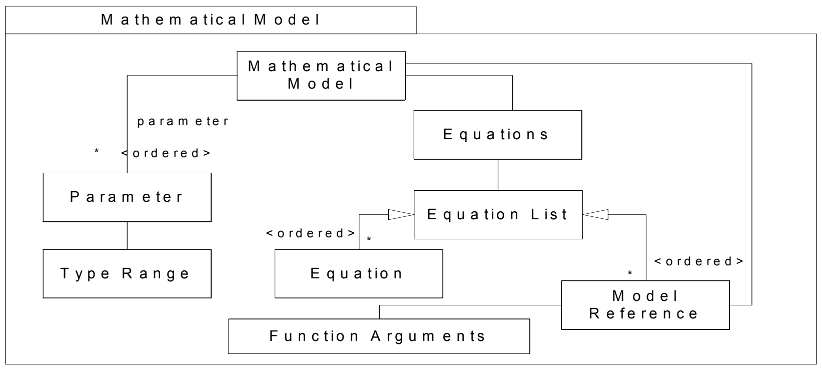
The Knowledge Base package within the Knowledge package describes the modelling of knowledge base that represents instances of knowledge. This package is shown in Figure 8.

Fig. 8. Knowledge Base Package



The Mathematical package within the Knowledge package describes the modelling of mathematical elements used in representing knowledge. This package is shown in Figure 9.

Fig. 9. Mathematical Model Package



7 Conclusion

Managing knowledge through knowledge-based systems is an important part of an enterprise's knowledge management initiatives. These systems have evolved from being stand-alone machines to being part of the enterprise's group of systems. The process of constructing KBSs is similar to other software systems with conceptual modelling playing an important rôle 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 could be adopted for knowledge modelling as well. While UML in its current state has its limitations, it is an extensible language and thus can be used to support the knowledge modelling activity through the profiles mechanism. Developing a profile is not an easy task and involves many steps. The next step in this research is to specify the well-formed rules and operations using OCL, then validate the profile using a UML compliant modelling tool and finally test real-life KBS requirements through case studies in a number of knowledge-intensive domains.

References

1. 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.
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. Studer, R., Benjamins, R.V., and Fensel, D.: *Knowledge Engineering: Principles and Methods.* Data & Knowledge Engineering, 1998. **25**: p. 161-197.
4. Angele, J., Fensel, D., Landes, D., Studer, R.: *Developing Knowledge-Based Systems with MIKE.* Journal of Automated Software Engineering, 1998. **5**(4): p. 389-418.
5. Grosso, W.E., Eriksson, H., Fergerson, R.W., Gennari, S., Tu, S., Musen, M.A.: *Knowledge Modelling at the Millennium (The Design and Evolution of Protege 2000)*. 1999, Stanford Medical Institute.
6. Booch, G., Rumbaugh, J., Jacobson, I.: *The Unified Modelling Language User Guide*. 1999, Reading, Massachusetts: Addison Wesley.
7. Fowler, M.: *What's a Model for?*, in *Distributed Computing Magazine*. 1999. p. 33-37. - Accessed at http://martinfowler.com/articles.html.
8. Fowler, M.: *Is There Such a Thing as Object-Oriented Analysis?* in *Distributed Computing Magazine*. 1999. p. 40-41. Accessed at http://martinfowler.com/articles.html.
9. Milton, N.: *Types of Knowledge Models*. 2002. Accessed at http://www.epistemics.co.uk/Notes/90-0-0.htm

10.Gomez-Perez, A., Benjamins,V.R.: *Overview of Knowledge Sharing and Reuse Components: Ontologies and Problem-Solving Methods.* in *IJCAI-99 Workshop on Ontologies and Problem-Solving Methods (KRR5)*. 1999. Stockholm, Sweden.

11.Gruber, T.R.: *Toward principles for the design of ontologies used for knowledge sharing*. 1993, Report KSL-93-04, Stanford University.

12.Kende, R.: *Knowledge Modelling in Support of Knowledge Management.* Lecture Notes in Atrifical Intelligence, 2001. **2070**: p. 107-112.

13.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.

14.Manjarres, A., Pickin, S., Mira, J.: *Knowledge model reuse: therapy decision through specialisation of a generic decision model.* Expert Systems with Applications, 2002. **23**(2): p. 113-135.

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

16.Protege,: *Protege Frequently Asked Question*. 2002. Accessed at http://protégé.stanford.edu/faq.html

17.Kingston, J. and A. Macintosh, *Knowledge management through multi-perspective modelling: representing and distributing organizational memory.* Knowledge-Based Systems, 2000. **13**: p. 121-131.

18.Chung, L., Subramaniam, N.: *Adaptable architecture generation for embedded systems.* Journal of Systems and Software, 2003. **17**(3): p. 271-295.

19.Kalogeropoulos, D.A., Carson, E.R., Colinson, P.O.: *Towards Knowledge-Based Systems in Clinical Practice: Development of an integrated Clinical Information and Knowledge management Support System.* Computer Methods and Programs in Biomedicine, 2003. **72**: p. 65-80.

20.Kobryn, C.: *A Standardization Odyssey.* Communications of the ACM, 1999. **42**(10): p. 29-37.

21.Steimann, F., Kuhne, T.: *A Radical Reduction of UML's Core Semantics.* Lecture Notes in Computer Science, 2002. **2460**: p. 34-48.

22.OMG: *Requirements for UML Profile*. 1999.

23.OMG: *Unified Modeling Language specification (version 1.4)*. 2001.

24.Perez-Martinez, J.E.: *Heavyweight extensions to the UML metamodel to describe the C3 architectural style.* ACM SIGSOFT Software Engineering Notes, 2003. **28**(3).

25.OMG: *MOF Specification version 1.4*. 2002.

26.OMG: *UML 2.0 Testing Profile specification*. 2003.

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

28.Clark, T., Evans, A., Sammut, P., Willians, J.: *Metamodelling for Model-Driven Development (draft): To be published.* 2003.