Modelling Knowledge-Based Systems Using UML Profile

Mohd Syazwan Abdullah, Ian Benest
Department of Computer Science
University of York
Heslington, YORK, YO10 5DD
United Kingdom
syazwan,idb@cs.york.ac.uk

Andy Evans, Chris Kimble
Department of Computer Science
University of York
Heslington, YORK, YO10 5DD
United Kingdom
andye,kimble@cs.york.ac.uk

***Abstract* – Knowledge-based systems (KBS) play an important rôle in managing an organisation's knowledge initiated through knowledge management processes. These systems are designed, and developed using knowledge engineering techniques that are similar to software engineering, but have more emphasis on the rôles of knowledge in the reasoning process. However, there are no modelling techniques available in this field and most of the techniques that are used are usually adapted from the software engineering domain. The Unified Modeling Language (UML) is a general-purpose language that can be extended to model requirements from domains that are not currently defined by it. One such extension mechanism is the UML Profile which extends the language at the meta-level without changing the main construct of the language. This paper presents an initial profile that can be used to design a KBS.**

I. INTRODUCTION

Knowledge management is becoming an increasingly important way of managing an organization's knowledge that is embedded in people, processes, information generators and customers. The need to manage knowledge in organisations has become the key factor for success in the knowledge economy. Organisations around the globe, are gearing up with knowledge management projects and strategies to harvest the value of knowledge in order to stay competitive and be innovative. The research in the field of knowledge management concentrates mainly on finding effective ways of managing knowledge through social and management perspectives; since knowledge resides in humans, it needs human techniques for its management. Managing knowledge is a human-oriented process, but technological tools such as knowledge-based systems (KBS) can be used in support for such initiatives.

Knowledge-based systems are widely used to manage knowledge from a systems-based approach. These systems are built based on different task types such as diagnosis, design, configuration design, assessment and so on, and are considered to be knowledge intensive tasks. The development process of a KBS is similar to that applied to any general system; 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].

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 describes what is a UML profile, while Section V presents the initial UML knowledge modelling profile constructed using identified modelling concepts. Section VI concludes this paper and indicates the direction for future work.

II. KNOWLEDGE-BASED SYSTEMS

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. The main difference is that in KBS it is assumed that knowledge is represented in some explicit form, and hence the increased importance of knowledge modelling [2].

KBSs are developed using knowledge engineering (KE) techniques [3] that are similar to software engineering (SE) techniques but have an emphasis on knowledge rather than data or information processing. KE advocates an engineering approach to the process of developing a KBS by emphasising conceptual modelling of the system at the design stage. Many KE methodologies have been developed using models; for example, CommandKADS [2], MIKE [4], Protégé [5], and KARL [4]. However, this has not always been the case. Previously KBSs have been built through the process of knowledge transfer [3] in which the knowledge of the expert has been directly transferred into the knowledge base in the form of rules. The transfer approach is limited in operational use as expert knowledge is hard coded within the system and this creates a new problem if the knowledge base is to be updated as changes require substantial effort [2] in reconstituting the coded rules. The shift towards the modelling approach has also enabled knowledge to be re-used in different areas of the same domain [3] and has resulted in reducing development costs [2].

III. KNOWLEDGE MODELLING

Knowledge modelling is used in knowledge acquisition activities as a way of structuring projects, acquiring and validating knowledge and storing knowledge for future use [6]. Knowledge models are structured representations of knowledge. They use symbols to represent pieces of knowledge and their relationships. Knowledge models are: (1) symbolic character-based languages – logic; (2) diagrammatic representations – networks and ladders; (3) tabular representations – matrices and frames and (4) structured text – hypertext. 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 representation techniques, 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.

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. 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.

A. Modelling Techniques

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

CommonKADS has become the *de facto* standard for knowledge modelling and is used extensively in European research projects. A suite of models is at the core of the CommonKADS methodology [2]. 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]. 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*" [7]. 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 also incorporated in CommonKADS for knowledge modelling purposes.

Recently it has become a trend for system developers and researchers in KE to adopt object oriented modelling in developing conceptual models for knowledge systems. This has been broadly studied and reported in [8-10]. 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.

B.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 like 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], Methodology and tools Oriented to Knowledge-based engineering Applications (MOKA) projects [10] and KBS developments in general as shown in the works of [8, 11, 12, and 16]. However, the main adopters of UML for knowledge modelling are CommonKADS [2] and MOKA [10]. 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 (KBE) 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 4.

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. To use UML for formally modelling knowledge-based systems, one suggested way is to enhance it by adopting the extension mechanism proposed by OMG known as profiles.

IV. PROFILE EXTENSION MECHANISM

The OMG [13] has defined two extension mechanisms for UML: profiles and meta-model extensions. Profiles are sometimes referred to as the "lightweight" extension mechanism of UML [14]. 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 meta-model that describes it. However, the usage of stereotypes is restricted as changes to the semantics, UML structure and the introduction of new elements to the meta-model are not permitted [15].

The "heavyweight" mechanism for extending UML is known as the meta-model extension which is defined through the Meta-Object Facility (MOF) specification [16] and which involves the process of defining a new meta-

model. Using this extension, new metaclasses and meta-constructors can be added to the UML meta-model. This extension is a more flexible approach as new concepts may be represented at the meta-model level. The difference between the profile and meta-model extensions comes from the restrictions on profiles in extending the UML meta-model [15]. These restrictions impose that profile based extensions must comply with the standard semantics of the UML meta-model. However, this restriction is not applicable to the MOF based extensions, which can define a new meta-model. The meta-model approach however, is also called a profile.

V. UML KNOWLEDGE MODELLING PROFILE

The scope of the profile is adapted from [17]. 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, which may 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 meta-modelling extension approach of UML. It is being designed with the following principles in mind:

- UML integration: as a real UML based profile, the knowledge modelling profile is defined based on the meta-model provided in the UML superstructure and follows the principles of UML profiles as defined in the UML 2.0.
- Reuse and minimalism: wherever possible, the knowledge modelling profile makes direct use of the UML concepts, extends them, and adds new concepts only where needed.

The main thrust in this section refers to the CommonKADS methodology for KBS development [2] and related discussion in [18]. Tasks are the main categorization of action that need to be performed by the KBS which typically refers to "what we want the system to do". Current studies on extending UML to model knowledge only concentrates on certain task types such as product design in MOKA [10] and UML-based product configuration design [8]. There are no specific studies being conducted in creating a generic profile that can be used for different task types; and research now underway at York is focusing on this work. A review and analysis of task types based on the literature [2] shows that the creation of a generic profile is possible if the extension used is defined in general terms with no reference to any task type or inference strategy when executing the task.

In [19] 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 profile. 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 [19].

The following knowledge modelling concepts have been identified from the literature [2], [18] and are itemised in Table 1. These concepts have been known for some years and they provide a firm foundation on which to base the model.

TABLE I: Knowledge Modelling Concepts

Modelling Concept	Description
Concept (class)	Class that represents the category of things
Inferences	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 rôle	Specifies the collection of domain knowledge that is used to make the inference
Dynamic knowledge rôle	Run-time inputs and outputs of inferences
Rule type	Categorization and specification of knowledge
Knowledge base	Collection of data stores that contains instances of domain knowledge types
Rule	Expressions about an attribute value of a concept

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 meta-model. Unless it is viewed as an extension of UML, it is not a profile, but just a plain meta-model. Efforts are currently focused on developing this meta-model 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 form the core of the knowledge modelling language and is shown in Figure 1 as the knowledge modelling profile.

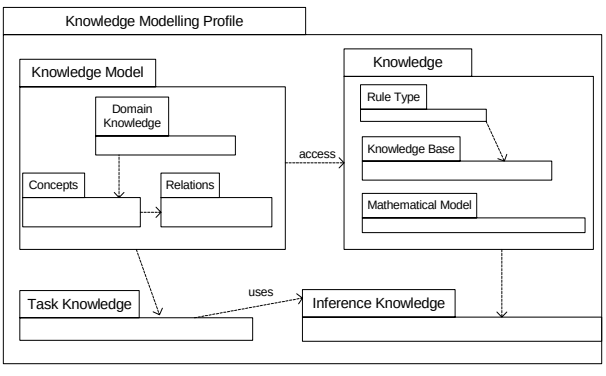

Fig. 1. Knowledge Modelling Profile Core Package

For reasons of space, this exposition concentrates on showing the concept, task knowledge, inference, rule type and knowledge base package aspects of the profile. The Concept package within the Knowledge Model package describes the concept of the profile. Concept here represents class. This package is shown in Figure 2.

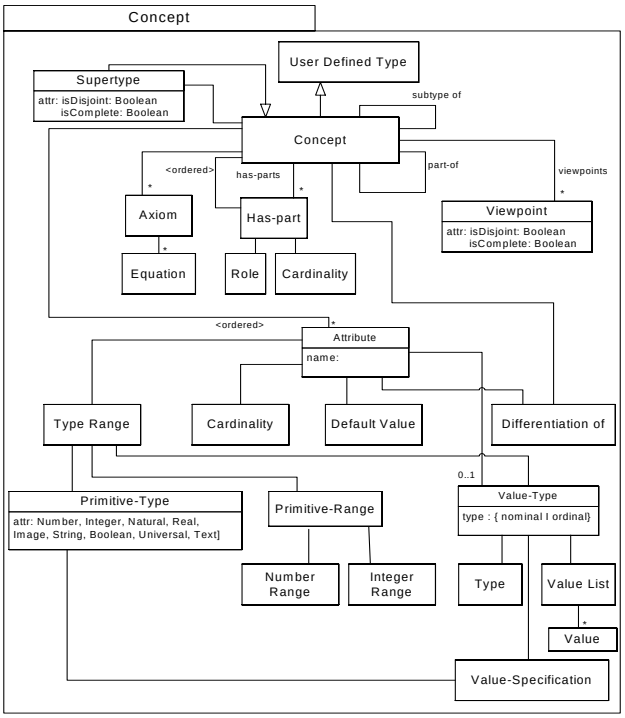

Fig. 2. Concept Package

The Task Knowledge package of the profile describes the task and task method in detail. This package is shown in Figure 3.

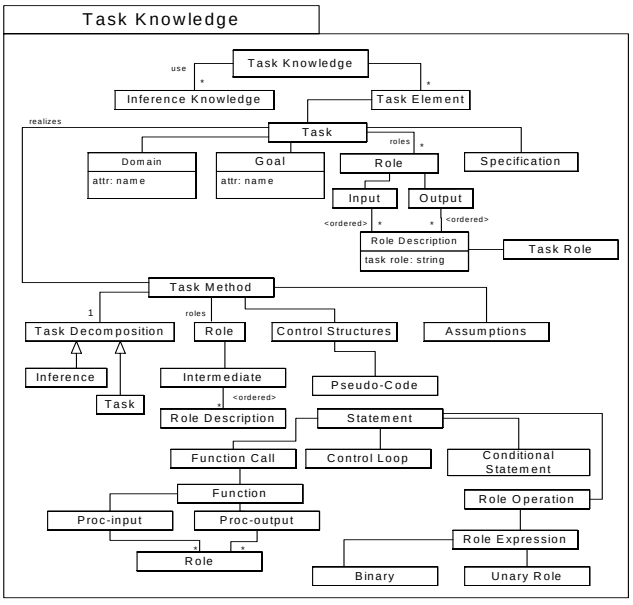

Fig. 3. Task Knowledge Package

The Inference Knowledge package of the profile describes the inference, knowledge rôle and transfer function in detail. This package is shown in Figure 4.

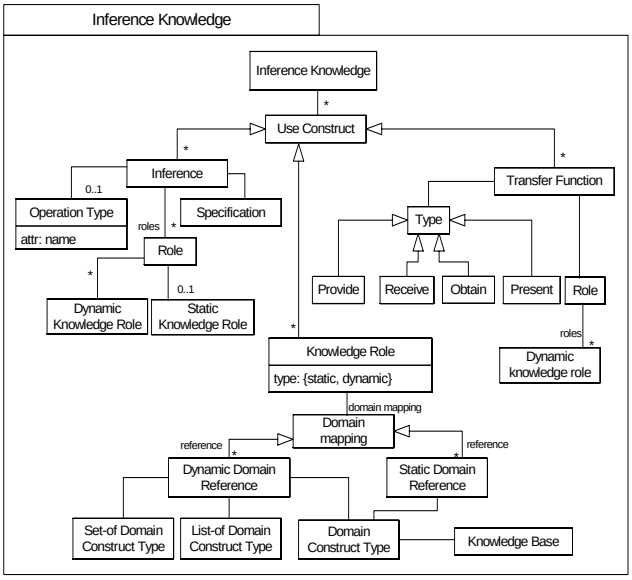

Fig. 4. Inference Knowledge Package

The Knowledge package of the core profile consists of three packages: the Rule Type package, the Knowledge Base package and the Mathematical Model package (not presented in the paper). The Rule Type package within the Knowledge package describes the modelling of rules. This package is shown in Figure 5.

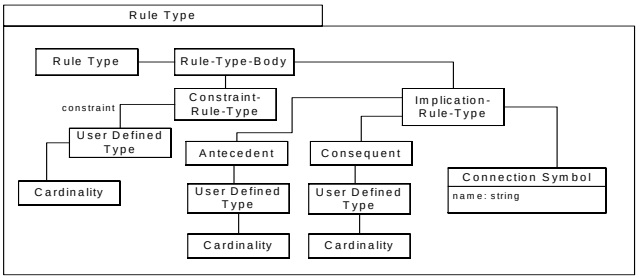

Fig. 5. Rule Type Package

The Knowledge Base package within the Knowledge package describes the modelling of the knowledge base that represents instances of knowledge. This package is shown in Figure 6.

Figure 7 shows part of the knowledge modelling profile used to represent the assessment task based on the template specified in the housing application case study discussed in [2]. A brief description of the case study is as 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 of 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. This example only concentrates on showing the abstracting process of the residence application (referred to as case-description which is a domain-independent term). 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, this allows a relatively large set of cases to be categorised. The example here is translating the original knowledge model (which is described in the textual knowledge modelling language of CommonKADS) to a UML object model based on the abstract syntax model shown earlier in this section.
The profile packages used here are the domain knowledge, inference knowledge and task knowledge. There are two concepts which represent the residence class and applicant class, and between these concepts a binary relation named residence application is created. For reasons of space, only one particular attribute related to each concept and its associated axioms are shown (in the upper part of the diagram in Fig. 7).

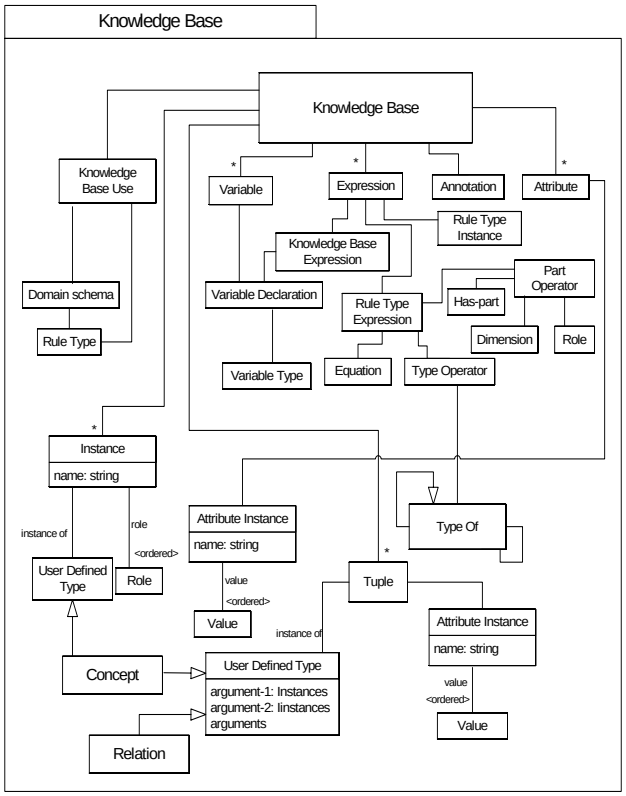

Fig. 6. Knowledge Base Package

The task knowledge package (shown as Task Knowledge in bold in Fig. 7) 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 inference “abstract”. The input for this task is “case descriptions” and the output will be “abstracted case”. The control structure will specify how the abstraction process will be performed by the task method. All this is shown in the lower part of the diagram. The inference knowledge package (shown as Inference Knowledge in bold) will carry out the reasoning process of abstracting the cases. The knowledge rôle 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 rôle. This knowledge is accessed from the knowledge base “Systems Description”. This is shown in the middle part of Fig. 7 on the right.

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 mechanism of the system being designed.

VI. CONCLUSIONS

Managing knowledge through knowledge-based systems is an important part of an enterprise’s knowledge management initiatives. 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 profile 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.

VII. ACKNOWLEDGMENT

The authors gratefully acknowledge the fellowship from Universiti Utara Malaysia in conducting this research

VIII. REFERENCES

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

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

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

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

[5] W.E. Grosso, H. Eriksson, R.W. Fergerson, J.H. Gennari, S. Tu and M.A. Musen, *Knowledge Modelling at the Millennium (The Design and Evolution of Protege 2000) SMI Report SMI-1999-0801*. 1999, Stanford Medical Institute.

[6] Milton, N., *Types of Knowledge Models*. 2002. Accessed at http://www.epistemics.co.uk/Notes/90-0-0.htm

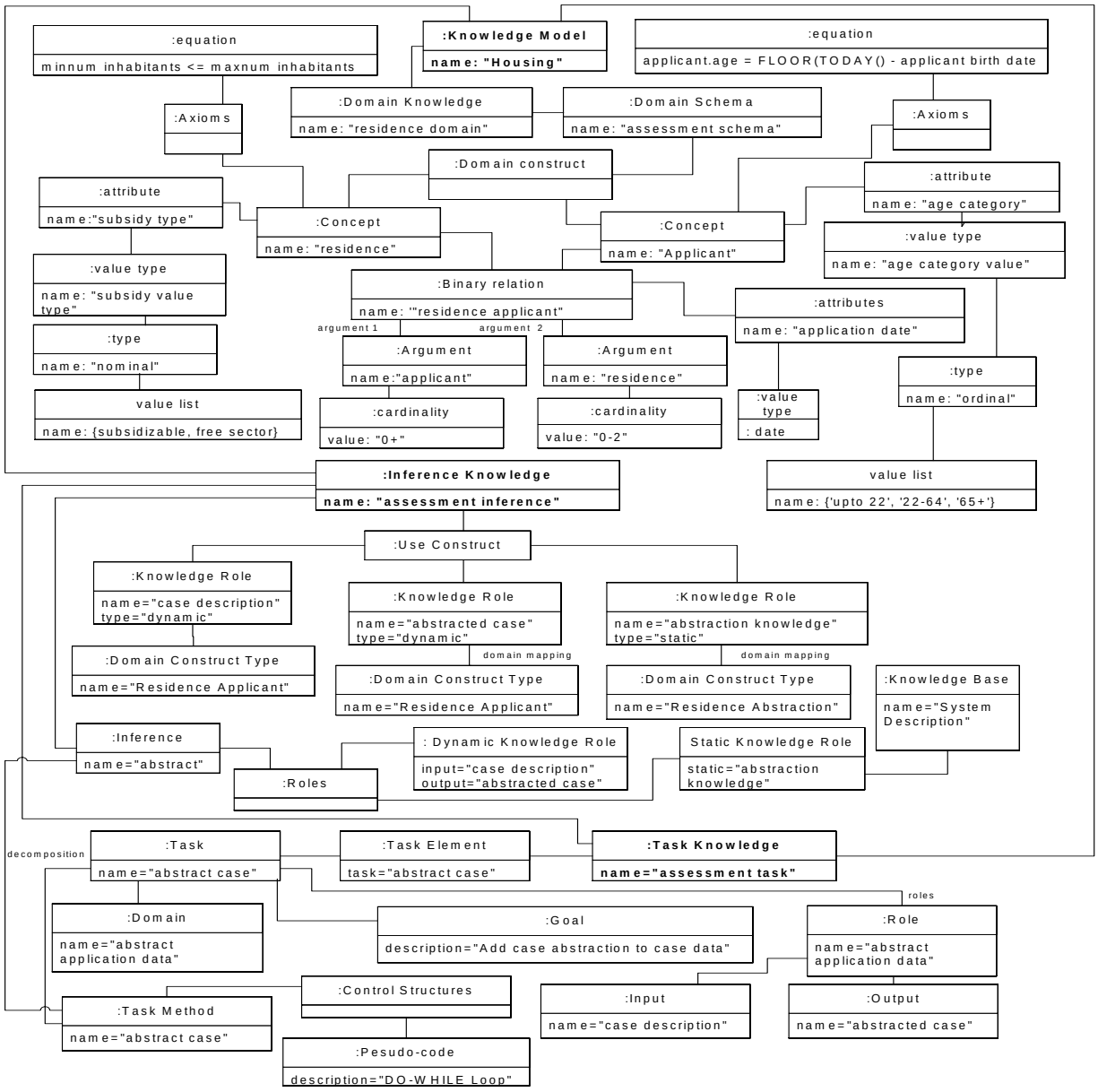

Fig. 7. Housing application assessment example

[7] Protege, *Protege Frequently Asked Question*. 2002. Accessed at http://portege.stanford.edu/faq.html

[8] A. Felfernig, G.E. Friedrich, and D. Jannach. *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.

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

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

[11] G. Simic and V. Devedzic, *Building an intelligent system using modern internet technologies.* Expert Systems with Applications, 2003. **25**: p. 231-246.

[12] A. Felfernig, G.E. Friedrich and D. Jannach, *UML as Domain Specific Language for the Construction of Knowledge Based Configuration Systems.* International Journal of Software Engineering and Knowledge Engineering, 2000. **10**(4): p. 449-469.

[13] OMG, *Unified Modeling Language Specification (version 1.4)*. 2001.

[14] OMG, *Requirements for UML Profile*. 1999, Object Management Group: Framingham, MA, U.S.A. p. 8.

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

[16] OMG, *MOF Specification version 1.4*. 2002.

[17] OMG, *UML 2.0 Testing Profile specification*. 2003.

[18] M.S. Abdullah, *Extending UML Using Profile for Knowledge-Based Systems Modelling*. 2003, Department of Computer Science, University of York: York.

[19] T. Clark, A. Evans, P. Sammut and J. Willans, *Meta-modelling for Model-Driven Development (draft): To be published.* 2004.