Using Unified Modeling Language for Conceptual Modelling of Knowledge-Based Systems

Mohd Syazwan Abdullah[1], Ian Benest[2], Richard Paige[2], and Chris Kimble[2]

[1] Faculty of Information Technology, Universiti Utara Malaysia (UUM),
06010 UUM-Sintok, Kedah, Malaysia
syazwan@uum.edu.my
[2] Department of Computer Science, University of York,
Heslington, York, YO10 5DD, United Kingdom
{idb,paige,kimble}@cs.york.ac.uk

Abstract. This paper discusses extending the Unified Modelling Language by means of a profile for modelling knowledge-based system in the context of Model Driven Architecture (MDA) framework. The profile is implemented using the eXecutable Modelling Framework (XMF) Mosaic tool. A case study from the health care domain demonstrates the practical use of this profile; with the prototype implemented in Java Expert System Shell (Jess). The paper also discusses the possible mapping of the profile elements to the platform specific model (PSM) of Jess and provides some discussion on the Production Rule Representation (PRR) standardisation work.

1 Introduction

Knowledge-based systems (KBS) were developed for managing codified knowledge (explicit knowledge) in Artificial Intelligence (AI) systems [1]. These were known as expert systems and were originally created to emulate human expert reasoning [2]. KBS are developed using knowledge engineering (KE) techniques [2], which are similar to those used in software engineering (SE), but they emphasise knowledge rather than data or information processing. Both KE and SE development processes have the same objective: to develop the system given the user requirements, in order to solve a particular problem related to the domain [2]. Systems development in SE involves the following iterative stages regardless of the methodology adopted: gathering and analysing user requirements, designing the system by translating user requirements into a software specification using conceptual models, coding the software specification into computer programs, testing the program to ensure the agreed results are produced, implementing the system and maintaining the system throughout its intended life span.

The KE processes for constructing a KBS in general are: requirements analysis involving identifying the scope for the KBS, designing the system by identifying the sources of expert knowledge for the KBS and how to represent them, acquiring the knowledge from the expert through knowledge acquisition techniques and constructing the knowledge base with instances of the domain knowledge, coding the system on target application languages or shells, testing the system to ensure the inference

C. Parent et al. (Eds.): ER 2007, LNCS 4801, pp. 438–453, 2007.

mechanism is working properly and producing the correct results, implementing the system incrementally and performing maintenance on the system [5, 26, 29]. In comparison with SE, the KE has one additional stage: that of knowledge acquisition (KA). This stage is vital in KBS development as the KBS is designed around the domain expert's knowledge of solving problems for a particular task, such as diagnosis, assessment and so on. The acquired knowledge is then used to populate the knowledge base in the form of rules, with which the system will perform reasoning. However, in SE there is no KA stage as the system is intended to capture information rather than reason with it and the actual dataset of the database will be populated by the system user when the system is deployed [26, 29]. Therefore, it may be concluded that the KA stage differentiates the SE and KE domains when developing software systems.

Central to this is the conceptual modelling of the system during the analysis and design stages of KBS development (known as knowledge modelling). A number of KE methodologies have emphasised the use of models, for example: CommonKADS, Model-based and Incremental Knowledge Engineering (MIKE), Knowledge Acquisition and Representation Language (KARL) and others [3]. 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 [4]. It has matured from a non-scalable technology [1, 5]. Once restricted to the research laboratory, it is now used for demanding commercial applications and is a tool widely accepted by industry [6, 7]. 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) [8] and knowledge-based engineering (KBE) services [9]. The standardisation of PRR is vital as it allows interoperability of rules between different inference engines – much needed by industry [10, 11].

The major problem with conceptual modelling of KBS (*known as knowledge modelling*) is that there is no standard language available to model the knowledge for developing a KBS. Most of the languages used are adapted from SE. The languages used in knowledge modelling are project based using a mix of notations such as Unified Modeling Language (UML), Integrated Definition Method (IDEF), Structured Analysis and Design Technique (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 [4] particularly if their designs were based on a standard language; it would help facilitate communication and sharing of blueprints among developers [12].

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 [13, 14]. Gill [13] 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. Conceptual models (CM) are a description of the

software system at different level of abstractions [15] and are popular in SE domain for providing an overview of concepts and relationships of the real-world, eliminate costly errors during analysis and design stages prior to construction and facilitates better communications between different people in the project team [16]. The importance of CM in software systems development are reflected through Model Driven Architecture (MDA) technique as models rather than codes have become the important artifacts of software development [17].

This paper is organised thus. Section 2 discusses the UML extensibility mechanicsm. Section 3 describes the knowledge modelling profile, and section 4 illustrates how the profile can be used to develop a KBS. Section 5 provides some discussion and finding on the use of the profile in PRR standardisation, while section 6 concludes with directions for future work.

2 UML, Model Driven Architecture and UML Profile Mechanism

UML is a general-purpose modelling language [18] that may be used in a wide range of application domains. Although UML is very popular and widely used as the modelling language for business applications, its use for knowledge modelling is limited. This is due to the fact that the usage of UML in modelling KBS has not been standardised [8], as there is no commonly agreed consensus on what KBS and KE concepts should be represented in a KBS design, and how rules should be defined and modelled. Nevertheless, there have been several attempts to use UML for knowledge modelling but such comprehensive efforts are only reflected in CommonKADS [26]. UML 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's Model Driven Architecture (MDA) – a model-driven engineering framework – provides integration with, and interoperability between, different models developed using its standards [18] (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 [18, 19]. Profiles have 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 knowledge modelling has concentrated only on certain task types such as product design and product configuration [20]. In contrast, the work described here emphasises the development of a generic profile for modeling the design knowledge of a KBS. Developing a meta-model for knowledge modelling 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. The OMG [21, 22] defines two mechanisms for extending UML: profiles and meta-model extensions. Both extensions have (unfortunately) been called profiles [18].

The "lightweight" extension mechanism of UML [22] is profiles. It contains a predefined set of Stereotypes, TaggedValues, Constraints, and notation icons that collectively specialize and tailor the existing UML meta-model. 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 [23]. The "heavyweight" extension mechanism for UML (known as the meta-model extension) is defined through the MOF specification [24] which involves the process of defining a new meta-model [23]. This approach should be favoured if the semantic gap between the core modelling elements of UML and the newly defined modelling elements is significant [18].

The work presented in this paper exploits the profile extension using the XMF (eXecutable Meta-modelling Framework) approach [25] as we believe that the knowledge modelling concepts can be modelled by tailoring existing UML meta-models without having to introduce new meta-concepts to UML. Furthermore, this will enable the profile to have readily available tool support, which will be a significant advantage for knowledge modellers in adopting UML over other languages. The OMG only specifies what profiles should constitute and not how to design them. By adopting the XMF approach, 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. This work uses the XMF approach in designing the profile and implementing it in the Mosaic tool. Although XMF core meta-model differs slightly from UML meta-model, and the same is true for Eclipse ECore meta-model, nevertheless the fundamentals are still the same. Furthermore, the knowledge modelling profile only extends the UML meta-class Class and Associations. However, only the profile concepts' extension to Class can be defined using Mosaic, as associations are implemented as built-in modelling features which are directly available to use at the model level.

3 UML Profile for Knowledge Modelling

The concepts for the knowledge modelling profile are re-used from the existing BNF definition of the CommonKADS Conceptual Modelling Language (CML) [26]; 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 [1, 27-29] and are widely used for representing concepts in KBS in the KE domain. The Knowledge Modelling profile was implemented using the XMF Mosaic by defining a meta-profile that allows for the definition of the knowledge modelling profile stereotypes, which in turn enables the construction of a knowledge model as an instance of the profile meta-model. To achieve this, the profile is defined as an extension to the XCore meta-model (the XMF-Mosaic's MOF based meta-model, similar to the definition of the UML meta-model) in the form of a meta-package for the profile. An important feature of the stereotypes is the inheritance of the modelling capabilities of UML meta-class elements. Meta-package is a mechanism in XMF-Mosaic that enables the content of the profile package to be viewed as an instance of the XCore meta-model class. The profile meta-model used here is the derived meta-model of CommonKADS and defined as the complete knowledge modelling abstract syntax meta-model in [32] as shown in Figure 1.

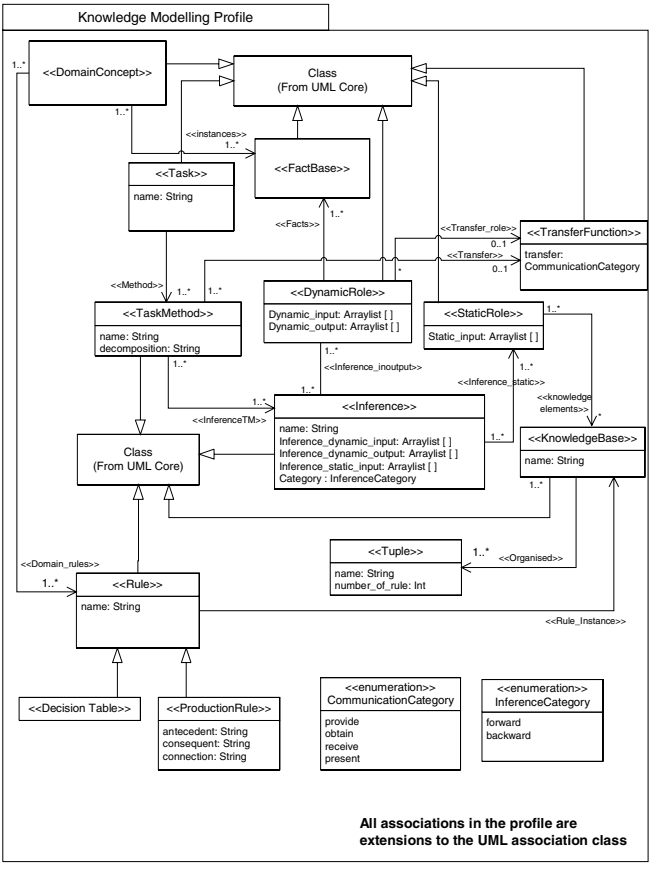

Fig. 1. Abstract syntax meta-model for knowledge modeling profile

The discussion of XCore meta-model here is related to implementing the profile in the XMF-Mosaic tool. Although the knowledge modelling profile meta-model is in UML, it is compatible with XMF-Mosaic because the elements that the profile extends are the standard MOF features in both tools. Furthermore, using various UML tools in implementing a UML profile is different as these tools have distinct implementation procedures or concepts in defining the profile in the tool, but this does not change the profile definition. Figure 2 shows the knowledge modelling profile stereotypes defined in XMF-Mosaic.

A Concept class is used to represent structural things and these have attributes contained in them; it is similar to class in the UML meta-model. When the attributes are used in rules they are known as knowledge elements. A Concept is linked to the Rule class in the model. Concepts are diagrammatically associated with FactBase; as the values of the attributes are stored here and are extracted during the reasoning process of the inference. The instances of each attribute, contained in the FactBase class, are accessed by the dynamic role, which passes them to the inference process that matches the premise with the consequent part of an implication rule.

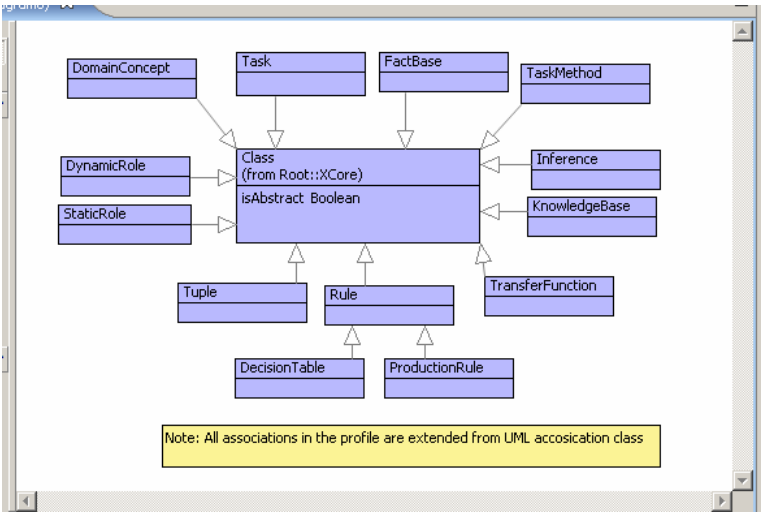

Fig. 2. Extension of the UML with stereotypes for the Profile

Task class defines the reasoning function and specifies the overall input and output of the task. Each task will have an associated task method that executes the task. The structure of the task, its task method, and the set of associated inference processes can be defined with the knowledge model from the problem-solving method library. The task-type, knowledge model, will help in identifying the inference structure needed to perform the desired task. Task method can be decomposed into sub-tasks for certain task-types. Task method class will specify the type of inference that is to be performed. The control structure of the method captures the inference reasoning strategy, which is described using an activity diagram. If the inference process requires additional input, either from the user or from an external entity, the task method will invoke a transfer function. Such functions are used to transfer additional information between the reasoning processes.

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 KBS 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 KBS 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 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 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.

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 two types of rule: implication rule, and decision table. 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 [10, 11] and is the focus of OMG's PRR [8] and KBE [9] standardisation work.

4 Case Study – The Clinical Practice Guidelines KBS

The purpose of this case study was to show the usefulness of the knowledge modelling profile in capturing the KBS requirements and to see the implementation value of the profile when building a KBS from scratch. To demonstrate that the profile is capable in bridging the gap between domain analysis and system implementation, a prototype KBS was built using the Java Expert System Shell (Jess) [29]. The possible mapping between the profile elements and Jess meta-model is also presented. The case study is based on the Clinical Practice Guideline (CPG) recommendations for managing patients with venous leg ulcers described in [30]. The CPG contains recommendations for assessment of ulcers patients, the management of treatment using compression therapy, cleaning and dressing of the ulcers, education and training of care through sharing of knowledge and quality assurance issues related to provision of leg ulcer care. Each of these categories is further divided into several related factors grouped together functionally. The guideline is evidence-based and these recommendations are gathered from systematic review reports complied by researchers in patient health care. The guideline contains recommendation statements, which were graded based on the following three strength of evidence: I- Generally consistent findings in a majority of multiple acceptable studies; II- Either based on a single acceptable study, or a weak or inconsistent finding in multiple acceptable studies; and III- Limited scientific evidence which does not meet all the criteria of acceptable studies of good quality.

4.1 Modelling and Development of Clinical Practice Guidelines KBS

The CPG recommendation was implemented as a KBS application for educational purposes to list the recommendations based on evidence strength using the following classification (a) evidence strength only; (b) evidence strength and category; (c) category

only; and (d) factors, evidence and category. The rules for the KBS was defined based on these classifications (in the actual recommendation, each recommendation has a brief explanation rather than ID as I1, II2, III4, etc which are much more convenient for discussions.).

The first stage in modelling KBS applications is to determine the nature of the problem [29] that the system should tackle and what the applicable task types available in the task catalogue are [2, 26]. The CPG can be regarded as a classification task, since the system classifies the recommendation based on four pre-defined criteria. To avoid any confusion, this task is referred to as a recommendation task, which is implemented using the task method 'match method', which consists of a single 'match' inference. This is shown in the task decomposition diagram in figure 3.

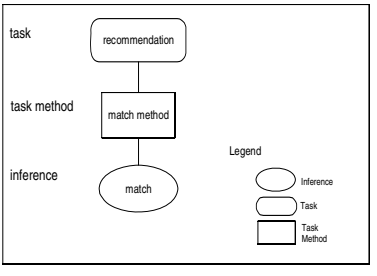

Fig. 3. Task decomposition diagram for CPG based on CommonKADS [26] notation

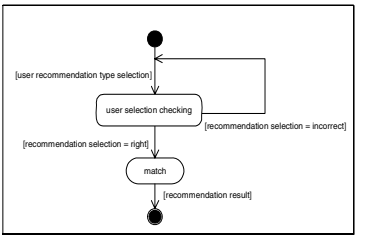

Fig. 4. CPG UML activity diagram

The control structure of this match method is shown using the activity diagram and is shown in figure 4. This is a straight-forward reasoning system as there are no loops in the recommendation matching process. The system user makes a recommendation type selection, and the resulting selection combinations are checked to ensure that they are valid. The selection is then matched with the recommendation value and the result is obtained. If incorrect selections are made, the selection process is repeated. Once the KBS task requirements and functionality have been determined, the knowledge model of the system is constructed using the knowledge modelling profile stereotypes. Most of the stereotypes of the profile were used, except for transfer function, as the CPG system does not need any input from external sources during the reasoning process and does not need any decision tables, as the rules for the system are represented by production rules. Figure 5 shows the knowledge model of the CPG application.

The KBS domain concept 'CPG' is composed of the five category of recommendations which are represented as domain concept 'CPGManagement', 'CPGCleansing', 'CPGQualityAssurance', 'CPGAssessment' and 'CPGEducation' shown at the top section of figure 5. Each of the domain concepts has three attributes (name, factors and evidence strength) upon which four types of rules for the system were defined based on their values. The instances of these attribute are stored in the fact base of the system which are accessed by dynamic role to get the facts for the inference reasoning process. The inference executes the reasoning task based on the task method specification which only specifies a single inference execution for the CPG system. The production rules of the system are stored in the knowledge base which are organised into tuples.

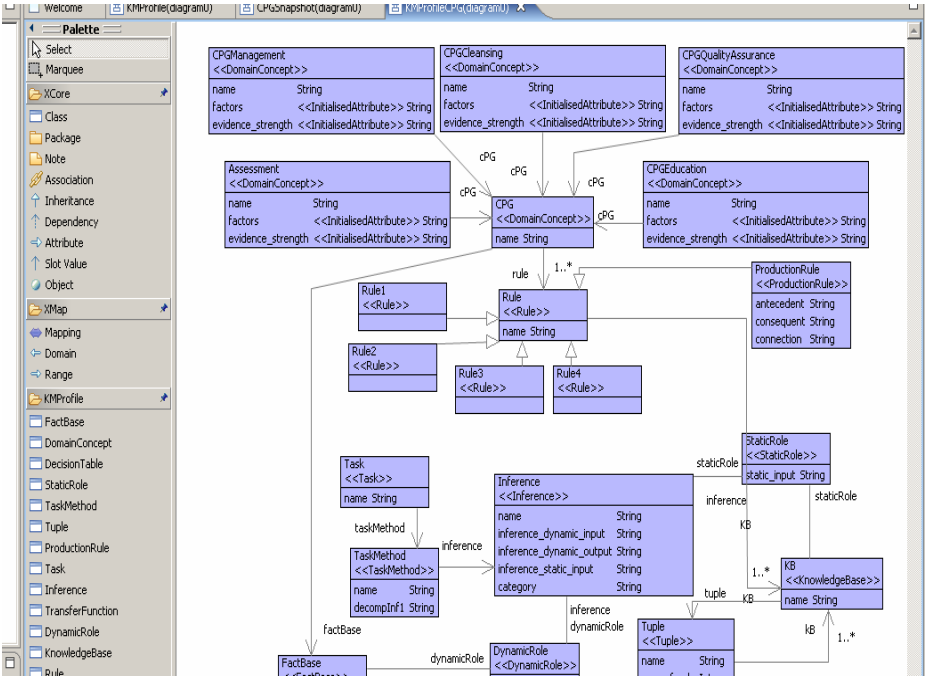

Fig. 5. CPG knowledge model

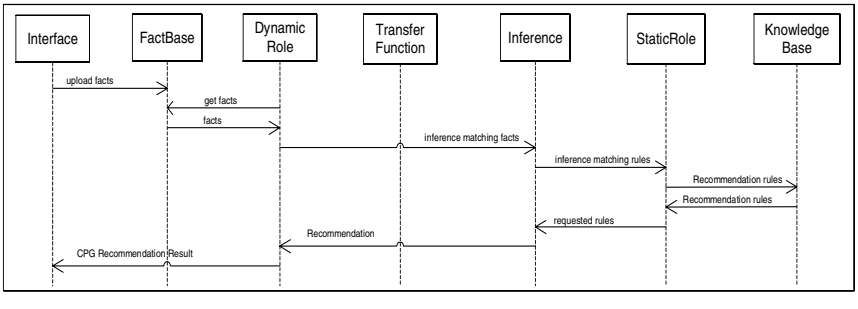

Fig. 6. Sequence Diagram of CPG system

KBS design is very much different to that of a conventional system, as the overall aim of the KBS is to gather the needed facts to fire the rules. In doing so, completing the whole reasoning cycle involves activation of different processes and message passing between objects. As a result, it is difficult to capture these vital information using object diagram due to the fact that several snapshots are needed to gather the whole picture. However, this limitation was solved with the aid of another type of UML diagram, namely the sequence diagram. Using sequence diagrams, the processing elements of the KBS gathered from the profile are listed as objects with an additional Interface object to model the flow of logic that captures the dynamic behaviour

of the KBS as shown on figure 6. The input from the user is entered through the interface which becomes the fact for the system when the recommendation type selection question has been answered. These facts are gathered by dynamic role and the inference engine gets these facts and matches them with the rule gathered from the knowledge base to provide the recommendation.

Table 2. Jess Program Summary for CPG System

```
;; Module MAIN
(deftemplate CPG) deftemplate S-C-F)
(deftemplate question)(deftemplate answer)
(deftemplate recommendation)
;;Module Question
(deffacts question-data)(defglobal ?*crlf* = "")
;; Module ask
(defmodule ask)(deffunction ask-user (?question
?type))(defmodule startup)
;; Module interview
(defmodule interview)
(defrule request-strength => assert ask strength)))
(defrule assert-user-fact
  (answer (ident strength)text ?i))(answer (ident
cate_gory)    (text ?d))(answer (ident factors_type)
(text ?j))=>  (assert (user (strength ?i) (cate_gory
?d)(factors_type ?j))))
;; Module recommend
(defmodule recommend)( defrule S-C-F-1-0-0
 user (strength ?i&:(= ?i 1))(cate_gory ?d&:
( = ?d 0))factors_type ?j&:(= ?j 0))) => assert
recommendation (S-C-F STR1) (explanation "Strength
equals 1 Recommendation ( I1 , I2 , I3 , I4 )") )))

;; Module report
```

The CPG prototype recommendation system was implemented using Java Expert System Shell (Jess) rule engine, which is a popular variation of the CLIPS rule engine developed in Java. Jess was chosen as the implementation platform as it is the reference implementation of the JSR 94 Java Rule Engine API that defines standard API for Java developer to interact with a Java rule engine widely used in commercial products and open source software projects.

The system receives the user input value for the strength, category and factor which are the facts for the system to fire the rules through the interview module based on the questions from the question module and the ask module performing error checking on the answers. In the recommendation module, the CPG rules are defined (evidence strength only; category only; evidence strength and category; and factors, evidence and category) and these rules are matched against the facts to fire the activated recommendation rule. The report module produces the recommendation report of the system which contains the explanation and the recommendation value. Table 2 presents portion of the Jess program summary for CPG system and the sample screenshot is shown in figure 7.

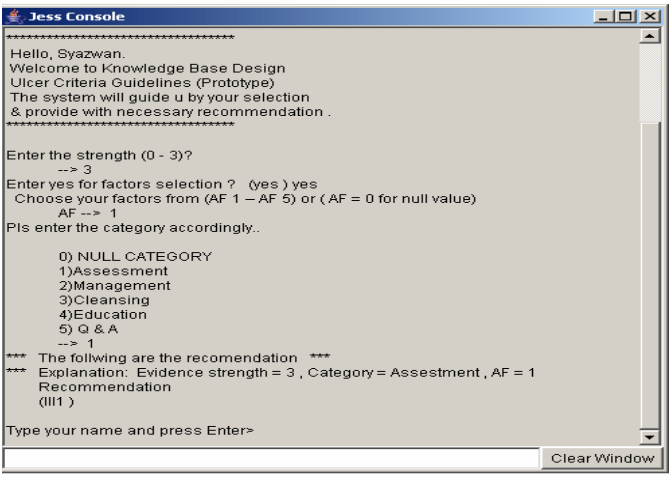

Fig. 7. Sample screenshot of the CPG system

4.2 Possible Mapping of the Profile to Jess

One of the key motivations for the MDA is in providing transformations between models (i.e. from a Platform Independent Model (PIM) such as a UML model or a profile model to Platform Specific Model (PSM) of a specific implementation platform such as Jess). The meta-model of Jess which defines the PSM is shown in figure 8. The purpose of this mapping is to translate a model of the profile into Jess implementation to prove that the profile is capable in bridging the gap between domain analysis and system implementation.

However, the profile meta-model elements cannot be directly mapped to all elements of the Jess meta-model and only partial mapping are technically possible. This limitation is due to the declarative nature of expert system shells programming and the need to have different level of abstraction between general KBS conceptual model and detail model of the implementation platform to enable model transformation in generating the specific program code. However, it is acknowledged that the knowledge modelling profile was very useful in understanding the KBS requirements for the CPG recommendations. This limitation is further discussed in detail on section 5.

Table 3 lists the possible mapping of the profile elements to the Jess. The domain concept elements of the profile can be mapped to `deftemplate`, `defclass` or `definstance` of Jess. However, for the CPG system, only `deftemplate` was used to represent the CPG domain concept which has three different slots for strength, factor type and category. The factbase element of the profile can be mapped to `deffacts` and for the CPG system; the question-data were used to gather the needed facts for the application. There are no direct mapping for task and task method to Jess but `defmodule` can be used to divide the application into structured modules. To perform the reasoning process, inference is activated through the function '`run`',

which is a Jess function that starts the pattern matching process. The dynamic role can be mapped to the Jess function '`assert`' which asserts all facts into the working memory of the inference engine. In the CPG system, this can be seen in the interview module in getting the facts to the working memory and asserting the recommendations.

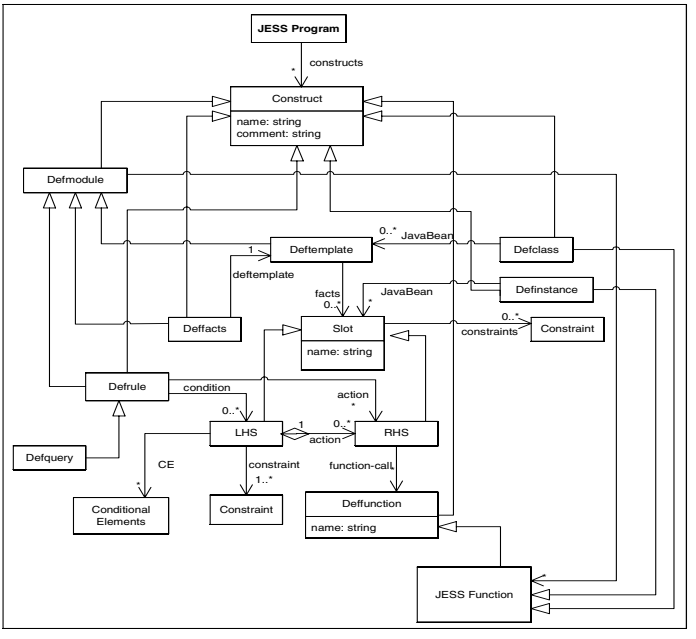

Fig. 8. Jess Meta-model

Table 3. Possible mapping of the Knowledge Modelling

Profile Concepts	mapping	JESS Concepts
DomainConcept	=	Deftemplate (Frame) Slot, Defclass Definstance
FactBase	=	Deffacts
Task	≈	Defmodule
Task Method	≈	Defmodule
Inference	≈	Deffunction – run ( )
Dynamic role	≈	Deffunction – assert ( )
Static Role	≈	Defmodule - focus
Transfer function	≈	Defunction
Knowledge base	≈	Defmodule - focus
Tuple	≈	Defmodule – focus (rules partition)
Rule	=	Defrule
• Implication Rule	=	Defrule - LHS, RHS
○ Antecedent	=	Deffunction, Conditional Elements
○ Consequent	=	Defquery

There is no direct mapping for knowledge base and tuple, but the `defmodule` constructs of Jess allows large number of rules to be physically organised into logical groups. Modules also provide a control mechanism that only allows the module that has the *focus* to fire the rule in it, and only one module can be in focus at a time. In the CPG system, the recommend module is used to organise the rules into knowledge base and static role can be mapped to the focus function of Jess since all the CPG rules for the inference engine are contained here. The role of transfer function in obtaining additional information can be mapped to the `defmodule` construct that implements the appropriate functions to get this information.

Table 4. CPG `'S-C-F-1-0-0'` rule

```
1 defrule S-C-F-1-0-0
2 user (strength ?i&:(= ?i 1))
3 cate_gory  ?d&:(= ?d 0))
4 factors_type ?j&:(= ?j 0)))
5 => assert recommendation S-C-F STR1) (explanation
6 "Strength equals 1 Recommendation I1,I2,I3,I4)"))))
```

The rule element of the profile can be mapped directly to the `defrule` construct of Jess in which the antecedent part corresponds to the left-hand side (LHS) of the rule and the consequent part corresponds to the right-hand side (RHS) of the rule. The following example of manual mapping the CPG system rule '`S-C-F-1-0-0`' shown in table 4 would help demonstrate this better.

In line `1`, we define the rule using defrule which states the name of the rule – in this case strength = 1, category = null and factor = null S-C-F-1-0-0 which will list all recommendation of strength values of 1. Line `2`, `3` and `4` is the LHS of the rule which consists of facts matching patterns and line `5` and `6` contains the function call (RHS) which asserts the recommendations values.

5 Discussions Related to OMG PPR Standardisation Work

The following discussions are intended to provide useful information regarding KBS modelling in the context of the OMG Production Rule Representation standardisation work. The PPR work mainly requires the use of activity diagrams to model the relationship between rulesets to action states. However, in this work we have identified that the use of activity diagram is limited to model a particular process of the system. Furthermore, class diagram can only provide partial snaphots of the system at a particular point in time which is less meaningful in complex inference cycles. To overcome this limitation, we have used the sequence diagram which clearly helps to understand the flow of logic in the system as shown in section 4.2.

The profile described in this paper would help in understanding how rules are related to the domain concept elements in the KBS and the processes that are involved in activating the rule to fire with the help of activity and sequence diagram.

Furthermore, the profile only shows the categories of rule which can be modelled in a single diagram with the other model elements. Thus the profile would help overcome the current problem of omitting rules from the model.

Mapping the profile to PSM is only limited to domain concept, factbase and implication rule. The rest of the profile elements are useful to describe the KBS and usually implemented differently as runtime concepts in various rule engines. Nevertheless, this proves that the most important work in designing and developing KBS is writing the rules based on the domain concepts which attribute values stored in the fact base will activate the rules. As such, the standardisation work in PRR should first emphasise on agreeing standard representation of rule elements in writing rules which are portable across different inference engines.

6 Conclusion and Future Work

This paper presented an extension to UML using the (lightweight) profile mechanism for knowledge modelling that allows the relevant structural properties of KBS to be represented at conceptual level. This allows knowledge models to be built using an object-oriented approach based on the standard modelling language that is widely adopted. The profile was implemented in an object-oriented meta-modelling language tool, XMF Mosaic that allows easier visual implementation of profile which diagrams are similar to the common UML editors.

The profile has been successfully tested on several case studies. This includes designs from scratch and re-engineering of existing KBS and the results are encouraging. Currently work has concentrated on building an Eclipse plug-in to support the profile as it is a popular implementation tool for UML profiles. The plug-in allows profile-compliant diagrams to be drawn and validated, and XML or XMI representations produced. The infrastructure in the Eclipse makes this mapping straightforward to implement. The future work in this area involves studying how to automate the generation of Jess code from the profile elements that can be mapped to Jess metamodel. The work in automating the generation of Jess code from models is still in a work in progress [31].

References

1. Giarratano, J.C., Riley, G.D.: Expert Systems: Principles And Programming. Course Technology, Boston, Massachusetts (2004)
2. Studer, R., Benjamins, R.V., Fensel, D.: Knowledge Engineering: Principles and methods. Data & Knowledge Engineering, 25(1), 161–197 (1998)
3. 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), Stockholm, Sweden (1999)
4. Ergazakis, K., Karnezis, K., Metaxiotis, K., Psarras, I.: Knowledge Management in Enterprises: A Research Agenda. Intelligent Systems in Accounting, Finance and Management 13(1), 17–26 (2005)
5. Awad, E.M.: Building Expert Systems: Principles, Procedures, and Applications. West Publishing, Minneapolis (1996)

6. Liebowtiz, J.: If you are a dog lover, build expert system; if you are a cat lover, build neural networks. Expert System with Applications 21, 63 (2001)
7. Preece, A.: Evaluating Verification and Validation Methods in Knowledge Engineering, in Micro-Level Knowledge Management. In: Roy, R. (ed.) Evaluating Verification and Validation Methods in Knowledge Engineering, in Micro-Level Knowledge Management, R, pp. 123–145. San Francisco, Morgan-Kaufman (2001)
8. Production, O.M.G.: Rule Representation - Request for Proposal, Object Management Group: Needham, USA. p. 57 (2003)
9. Services, O.K.: for Engineering Design - Request for Proposal, Object Management Group: Needham, MA, US. p. 32 (2004)
10. McClintock, C.: ILOG's position on Rule Languages for Interoperability. In: W3C Workshop on Rule Languages for Interoperability, Washington, D.C, USA (2005)
11. Krovvidy, S., Bhogaraju, P., Mae, F.: Interoperability and Rule Languages. In: W3C Workshop on Rule Languages for Interoperability, Washington, DC, USA (2005)
12. Abdullah, M.S., Benest, I., Evans, A., Kimble, C.: Knowledge Modelling Techniques for Developing Knowledge Management Systems. In: Abdullah, M.S., Benest, I., Evans, A. (eds.) 3rd European Conference on Knowledge Management, Dublin, Ireland (2002)
13. Gill, G.T.: Early Expert Systems: Where Are They Now? MIS Quarterly 19(1), 51–81 (1995)
14. Tsui, E.: The role of IT in KM: where are we now and where are we heading. Knowledge Management 9(1), 3–6 (2005)
15. Juristo, N., Moreno, A.M.: Introductory paper: Reflections on Conceptual Modelling. Data & Knowledge Engineering 33(2), 103–117 (2000)
16. Dieste, O., Juristo, N., Moreno, A.M., Pazos, J., Sierra, A.: Conceptual Modelling in Software Engineering and Knowledge Engineering: Concepts, Techniques and Trends. In: Chang, S.K. (ed.) Handbook of Software Engineering & Knowledge Engineering, pp. 733–766. World Scientific Publishing, Hackensack, NJ (2002)
17. Jézéquel, J.-M., Hussmann, H., Cook, S.: A Metamodel for the Unified Modeling Language. In: Jézéquel, J.-M., Hussmann, H., Cook, S. (eds.). UML 2002. LNCS, vol. 2460, Springer, Heidelberg (2002)
18. Muller, P.-A., Studer, P., Bezivin, J.: Platform Independent Web Application Modeling. In: Stevens, P., Whittle, J., Booch, G. (eds.). UML 2003. LNCS, vol. 2863, Springer, Heidelberg (2003)
19. Brown, A.W.: Expert's voice - Model driven architecture: Principles and practice. Software and Systems Modelling 3(4), 314–327 (2004)
20. Abdullah, M.S., Kimble, C., Paige, R., Benest, I.: Developing UML Profile for Modelling Knowledge-Based Systems. In: Aßmann, U., Aksit, M., Rensink, A. (eds.) MDAFA 2003. LNCS, vol. 3599, Springer, Heidelberg (2005)
21. OMG. UML 2.0 InfrastructureFinal Adopted Specification, [cited 2004 5 April], Available from (2003), http://www.omg.org
22. OMG, Requirements for UML Profile. 1999, Object Management Group: Framingham, MA. p. 8.
23. Perez-Martinez, J.E.: Heavyweight extensions to the UML metamodel to describe the C3 architectural style. ACM SIGSOFT Software Engineering Notes, 28–3 (2003)
24. OMG. MOF Specification version 1.4. 2002 [cited 2004 5 April], Available from, http://www.omg.org
25. Clark, T., Evans, A., Sammut, P., Willians, J.: Metamodelling for Model-Driven Development (draft) (To be published 2004), http://albini.xactium.com

26. Schreiber, G., Akkermans, H., Anjewierden, A., De Hoog, R., Shadbolt, N., De Velde, W.: Knowledge Engineering and Management: The CommonKADS Methodology. MIT Press, Massachusetts (1999)
27. Cuena, J., Molina, M.: The role of knowledge modelling techniques in software development: a general approach based on a knowledge management tool. International Journal of Human-Computer Studies 52, 385–421 (2000)
28. Håkansson, A.: UML as an approach to Modelling Knowledge in Rule-based Systems. In: The Twenty-first SGES International Conference on Knowledge Based Systems and Applied Artificial Intelligence (ES2001), Peterhouse College, Cambridge, UK (2001)
29. Friedman-Hill, E.: Jess in Action: Rule-Based System in Java. Manning Publications, Greenwich, US (2003)
30. Clinical, R.C.N.: Practice Guidelines: The management of patients with venous leg ulcers. Royal College of Nursing Institute, London (1998)
31. Wu, C.G. (2004) Modelling Rule-Based Systems with EMF. Accessed at http://www.eclipse.org/articles
32. Abdullah, M.S., Profile, A U.: for Conceptual Modelling of Knowledge-Based Systems, Unpublished PhD Thesis, University of York (2006)