An Ontology-Based Knowledge Model for Software Experience Management

Abdulmajid H. Mohamed[a], Sai Peck Lee[b], Siti Salwah Salim[c]

Faculty of Computer Science and Information Technology
University of Malaya,
Lembah Pantai, 50603 Kuala Lumpur, Malaysia.
[a]abdlmajid@perdana.um.edu.my [b]saipeck@um.edu.my [c]salwa@um.edu.my

Abstract

The efficient management of experience knowledge is vital in today's knowledge-based economy. This paper is concerned with proposing a model for software experience knowledge. The model is aimed to represent the backend of knowledge management tools that support organisational learning activities in a typical software organisation. Generic knowledge models have failed to produce good quality software experience management tools. The diversity in the types and forms of software experience knowledge makes it necessary to establish customised knowledge models to effectively accommodate such domain-specific knowledge. Our model extends the simple generic knowledge models that usually rely on non parameterised knowledge (i.e. free text) or that just reflect certain types of software experience knowledge (tacit or explicit). In contrast, our model aims to strike a delicate balance between explicit and tacit knowledge that in a way act as correlated information sources. This model can be easily adapted to software-oriented knowledge management tools eliminating many limitations of existing knowledge models used for the same purpose.

Keywords: Knowledge management, software experience knowledge, tacit Knowledge, ontologies, knowledge model.

1. Introduction

Knowledge modelling is often considered as the first step in developing Knowledge-Based Systems (KBS). The aim of this process is to understand the types of data structures and relationships within which knowledge can be held, and reasoned with.

We focus our attention on software experience modelling. For the course of software experience management, this involves explicitly defining structures that can be used as templates to store software experience knowledge (i.e. knowledge model). The description of the knowledge model should include the artefact types as well as the relationships between these artefacts. The information retrieval is then accomplished based on the defined relationships and the chosen indexing criteria (for example, ontology, CBR, hypertext, etc.).

One of the crucial decisions to be made when building knowledge management (KM) solutions is the characterisation of what are the knowledge fragments to

consider. Figure 1 depicts the different ways to classify organisational knowledge. According to the KM literature, the organisational knowledge can be categorised as tacit and explicit [1]; or individual and group knowledge [2]. In the former category, tacit knowledge refers to knowledge which is not explicitly captured. In other words, it relates to organisational undocumented knowledge. Tacit knowledge is usually held in individual's memories in the form of perceptions, beliefs, viewpoints, know-how, etc. The significance of this type of knowledge lies in that organisations have no control on its usage and lifetime, yet it could represent a major threat against an organisation's business interests. For example, such knowledge can easily fall in the hands of competitors as a result of frequent staff turnover. As such, a major challenge in KM research is how to recognise, generate, share and manage tacit knowledge. On the other hand, explicit knowledge refers to any knowledge that can be documented, archived and codified. This includes plans, business documents, guidelines, process models, etc.

Organisational knowledge can also be categorised as individual and group-based knowledge. Individual knowledge is largely realised in a tacit form (i.e. in workers' heads). However, it may also be represented explicitly in a semi-structured way (e-mails, personal notes, etc.) but the scale of this knowledge is very small compared to the individual tacit knowledge. On the other hand, group-based knowledge refers to the collective knowledge related to individuals interacting in team-based work.

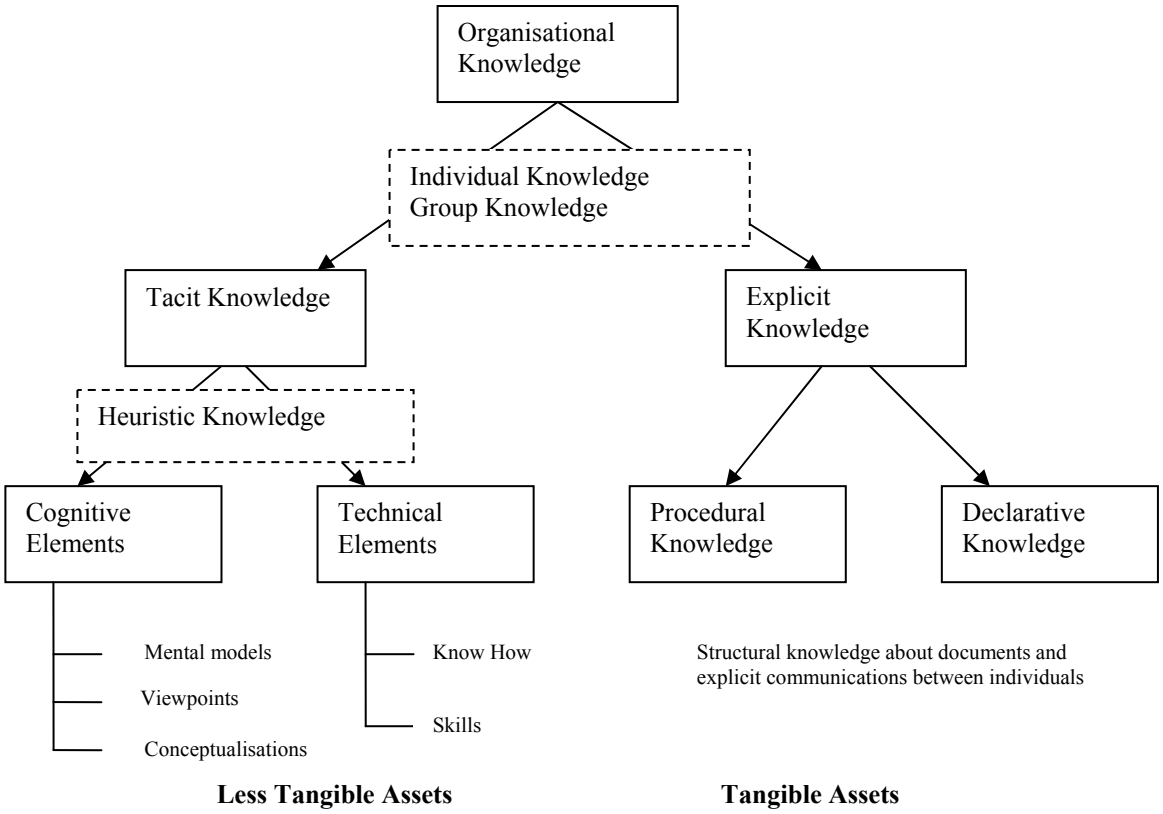

Figure 1: Theoretical Knowledge taxonomy [15]

2. Software Knowledge Management

Traditionally, knowledge creation and exchange in software organisations is communicated through natural language either verbally or vocally. Verbal knowledge is usually presented in plain text augmented with diagrams and software engineering notations. Some of this knowledge is stored electronically while others may be kept as hard-coded documents (for example, personal notes). The knowledge documented electronically is stored in various formats processed by different tools (i.e. word processors, drawing software, project management tools, and CASE tools).

It is believed that good software documentation would help software developers make good decisions in upcoming projects. However, in spite of strict documentation policies imposed by some software organisations, there is a type of knowledge that is hardly captured which is the tacit knowledge. For instance, a huge part of the details during a meeting are unrecorded and only resided in the developers' minds. This limitation deprives the organisation of very important information. This includes assumptions, alternatives and views behind software decisions taken. The Rationale Management is introduced as one of the software engineering topics that tackle this issue. It aims to 'improve the quality of decisions by making decision elements, such as criteria, priorities, and arguments explicit [3]. Developers of subsequent projects can view the organisational experience in the form of past software decisions from they can learn.

3. Our Proposed Knowledge Model

Before describing the components of our software knowledge model, we firstly describe our notion of what constitutes software knowledge fragments as bases to describe the varieties of software knowledge assets. The proposed knowledge model is then tailored on such characterisation of software experience knowledge.

3.1 Characterisation of Software Knowledge Assets

Unlike information management systems where all aspects of organisational data are considered, in knowledge management systems, the focus should be on knowledge fragments rather than information fragments. Knowledge fragments can be defined as the knowledge pieces that have been proved useful through experience. These fragments are created as a result of intensive and critical communications between respective knowledge workers. In other words, organisational knowledge is the organisational information enriched with different criteria and assumptions that represent context within which that knowledge was created.

For the course of determining the basic building blocks of our model, the model is built on the notion of ***K*-Asset** (**K**nowledge **Asset**) as the smallest granularity in the software experience knowledge. Basically, we regard the *K*-Asset as any useful proven fragment of software development knowledge. Any lesson learned or knowledge-embedded software artefact can be considered as candidate *K*-Asset regarding that it fulfils certain domain qualities. As shall be described in Section 3.2, K-Asset elements include what we regard as reusable artefacts in the software engineering process. These artefacts are not restricted to reusable software artefacts in the form of software code or libraries, but also involve artefacts in the form of know-how and recommended modelling or development tools.

3.2 Model Components and the Realisation Software K-Assets

Basically, our model can be seen in analogy to the data models of database management systems. It is used to guide knowledge generation and sharing in respective software-oriented KM tools. Specifically it guides tool users while they are populating or retrieving knowledge fragments. Figure 2 represents the Meta model describing the proposed knowledge model as a high-level representation. It acts as the domain ontology that describes different constituent ontologies used to symbolise basic ingredients of the experience drawn from software production line.

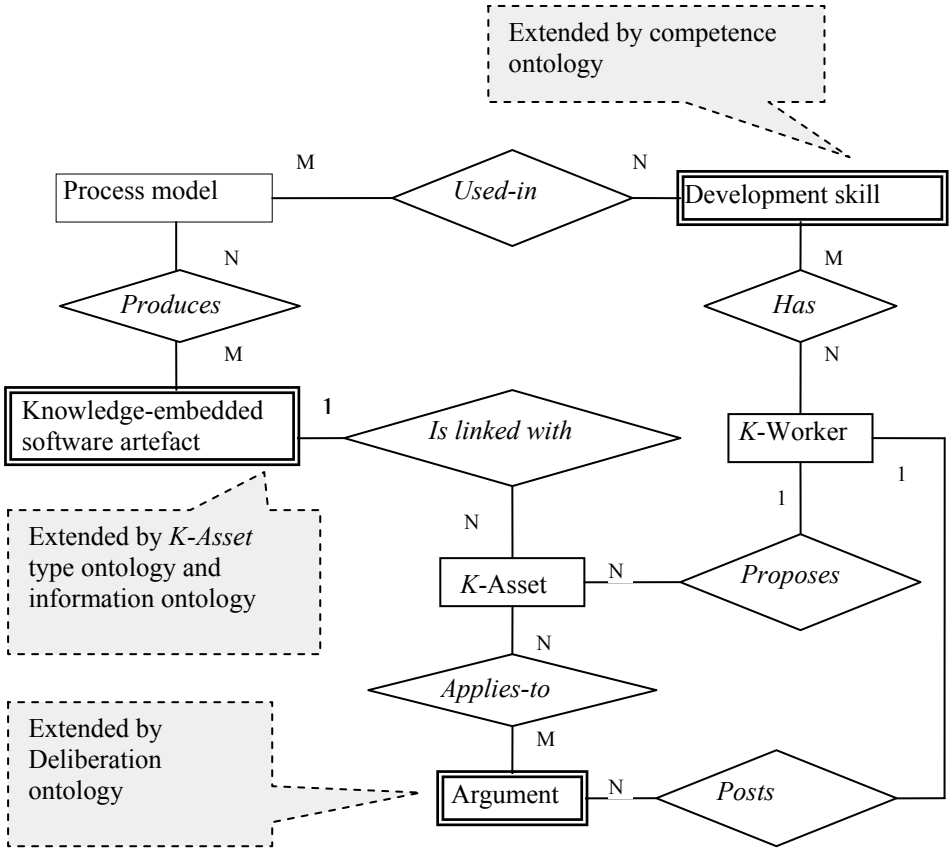

Figure 2: A higher-level software knowledge model (domain ontology)

As it is cited by Conklin, the biggest barrier to knowledge sharing is the "lack of shared understanding, especially about key concepts and terms" [4]. Research about ontologies aims to overcome this limitation. According to Vasconcelos et al [2], ontology is a "*formal and explicit specification of a shared conceptualisation"*. It symbolises the entities and relationships that define any particular domain (for example, software engineering). Having developed the domain ontology, all potential *K*-Assets are linked to the defined ontologies (semantic annotations). The same ontologies will be used later to search through the mass of *K*-Assets held in the resultant knowledge repository.

Based on our knowledge model (i.e. higher-level domain ontology), an individual *K*-Asset is described by four types of ontologies: (i) ***competence ontology***; (ii) ***information ontology***; (iii) ***type ontology*** and (iv) ***history ontology***. The ***competence ontology*** is used to categorise respective *K*-Assets based on a topic map that defines software competencies. The ***information ontology*** illustrates the attributes used to describe any *K*-Asset contents. Different attributes are used to characterise different *K*-Assets based on the *K*-Asset types represented by the *type ontology*. Attributes are filled in by the author of any *K*-Asset before being submitted to the knowledge repository. Figure 3 represents the information ontology (i.e. attributes) of a *K*-Asset characterised as a *lesson learned*.

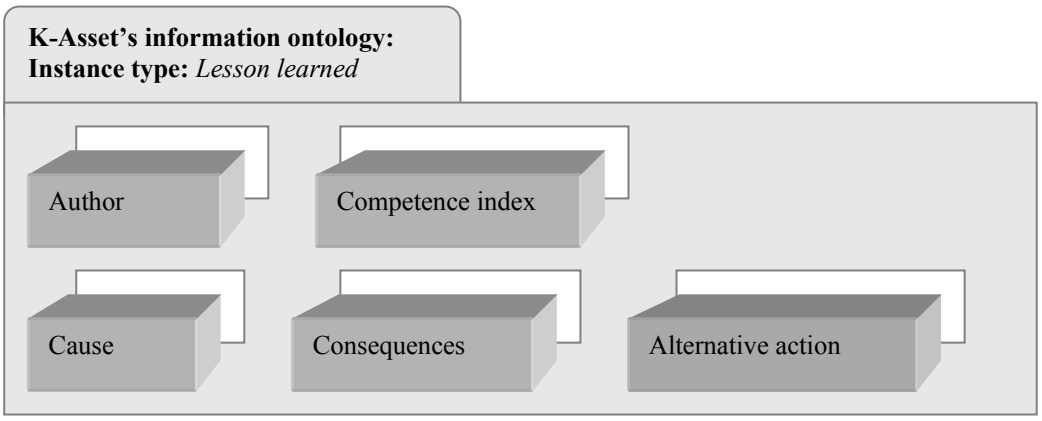

Figure 3: An Information Ontology for a Lesson Learned K-Asset Type

The ***type ontology*** represents the *K*-Asset types based on the defined types of the knowledge-embedded software artefacts. Figure 4 depicts candidate *K*-Assets as represented by the *type ontology*. They include *process models*, *software artefacts,* and *lessons learned* drawn from the software development process. Software artefacts include *data models, test suites, screen shots, tables, tool recommendations, code and functional diagrams*. We regard these types of artefacts as the salient by-products of the software development process.

In this era of COT-based software development which is basically code-based reuse, reusing functional diagrams, data models and other know-how information has become a necessity. For example, as a result of the recent diffusion of e-business applications, similar scenarios are likely to be adopted frequently, be it in the form of functional modelling level or the object modelling level. We additionally regard development tools as valuable reusable artefacts especially the ones that excel in the task for which they are designed. In this regard, we view the *development tool* node at the type ontology as a generic node where any type of programming, drawing, compiling, modelling or editing tool can be recommended as a valuable K-Asset.

Finally, the *lesson learned* node represents descriptions of what could be considered by developers as lessons learned. Each lesson can be thought of as an avoidable negative practice. Each K-Asset characterised as a lesson learned includes descriptions like the causes of the problem, its symptoms and alternative actions that could be taken to avoid the lesson reoccurrences.

Process models includes any *process descriptions* or *installation procedures* or *bug workarounds*. These types of K-Assets also represent a major source for learning the skills and know-how. This includes describing the know-how of any installation or programming or modelling process.

The *competence ontology* (see Figure 5) is mainly used as an indexing schema for all *K*-Asset types. This categorisation is based on the basic skills and competences of the domain. Instances of this ontology are arranged as taxonomy of domain competences represented as *is-a* hierarchy and *part-of* relation similar to the Object-Oriented (OO) structuring of elements. The OO-like hierarchy is used in order to utilise the inheritance rules to recall similar or partial results to user queries.

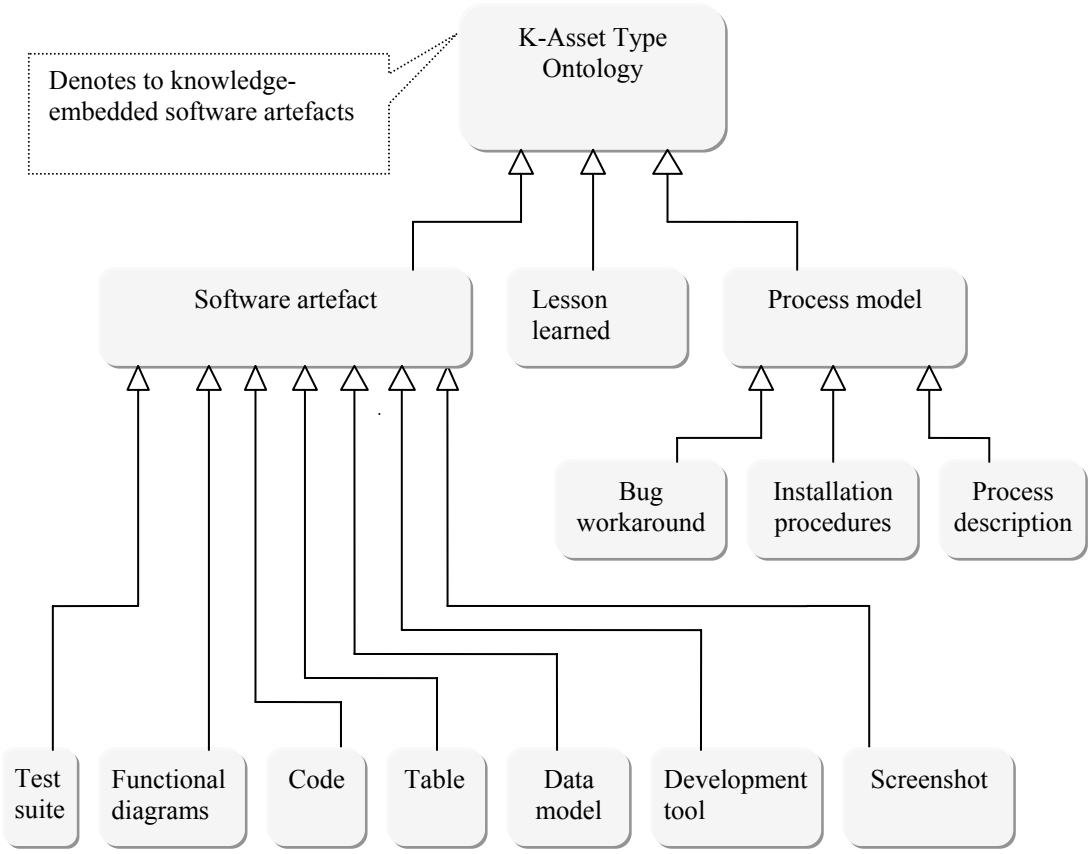

Figure 4: K-Asset's Type Ontology

Since *K*-Assets are usually created or modified in a collaborative manner, any knowledge generated as the outcome of such collaborative knowledge filtering has to be captured as well. Capturing this type of knowledge (i.e. decision rationale) shall be the responsibility of the ***history ontology***. This ontology includes information related to rationale behind active *K*-Assets. This part is the most important part as it plays the main role of capturing tacit knowledge. It describes the rationale attached with constituent *K*-Assets. The details of rationale are represented by an IBIS-based deliberation model that we proposed in [5]. Components of this model are shown in Figure 6.

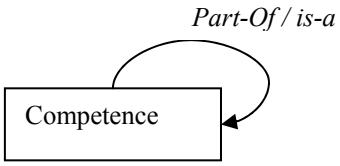

Figure 5: Competence Ontology

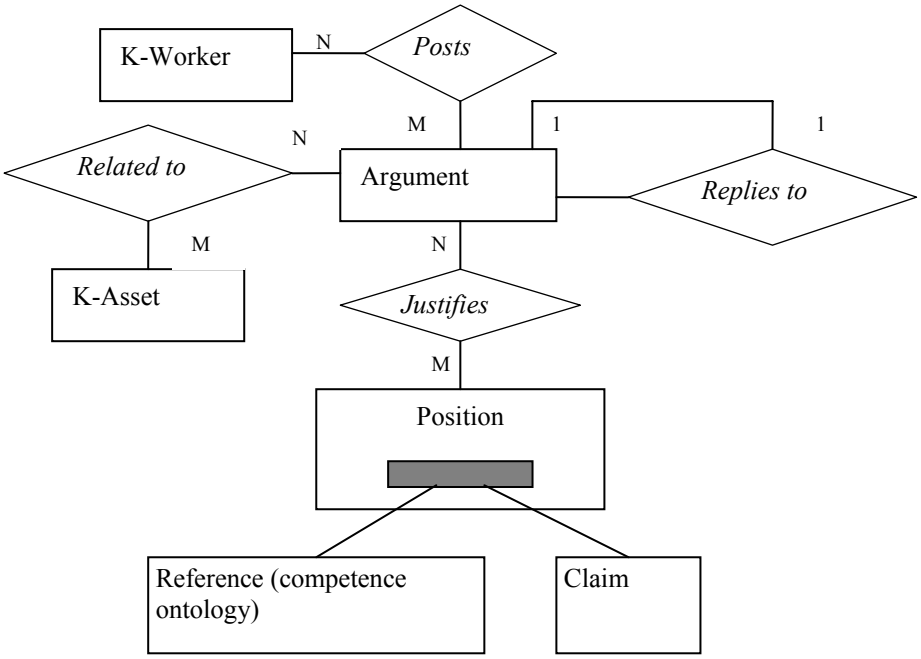

Figure 6: The proposed IBIS-based argumentation model (Deliberation ontology)

3.3 Intended use of the model

Designers of software-oriented KM tools will have a strong motivation to use our knowledge model in recording and retrieving experience knowledge. This should also be as part of a framework that governs KM activities in a software organisation. The model's knowledge taxonomy that comprises all types of integrated ontologies enable representing experience knowledge as a semantic net through which conceptual search can be employed. In addition to keyword-based search, tools that adopt this knowledge model can employ an ontology-guided search to generate fuzzy and non-zero hit queries while searching the K-Asset repository. In other words, the model enables KM tools to retrieve not only *K*-Assets that match a particular node at the competence ontology. Since the instances of the competence ontology are structured in OO-like hierarchy, inheritance rules can be used to include generic nodes to retrieve *K*-Assets similar to the target ones. For example, based on the instance of the competence ontology

shown in Figure 7, instead of limiting the search through *K*-Assets annotated as ***PHP*** scripts, the node ***web programming*** is selected, and then all *K*-Assets annotated as web programming tools shall be considered among which is PHP scripts.

4. Related Research

Efforts such as, REMAP [6], TeamInfo [7], Answer Garden [8], QuestMap [14], Designer Assistant [9], REFSENO [10]) and BORE ([11]), [12]) can be regarded as the main research stream that contributes to software knowledge management. However, the knowledge models employed by these approaches vary. TeamInfo's knowledge model relies on representing semi structured data in the form of E-mail messages. BORE relies on representing software knowledge in a structured way using the Case-Based Reasoning (CBR). Each case is described by fields such as: description, solution, characteristics, owner, etc. QuestMap's knowledge model also relies on semi structured representation of the knowledge in the form of nested YES/NO branching of questions and answers. Answer Garden's knowledge model represents software knowledge as a network of multiple-choice questions and answers. Designer Assistant's knowledge model represents the captured knowledge in the form of *advices*. An advice is composed of one to two simple statements that are given as a tool response to users' enquiries.

REMAP and REFSENO are the closest efforts to our approach. REMAP also installs IBIS as an embedded component similar to our knowledge model, but our model extends REMAPS characterisation of what is considered to be a software knowledge asset. Since REMAP is only aimed to capture software organisation's knowledge particularly in the requirement analysis phase, we believe it only captures limited proportion of software experience knowledge, because the software knowledge assets span all phases of the software lifecycle. It even expands to the post installation in the form of customer suggested knowledge assets. REFSENO's knowledge model is also similar to ours in that it also relies on structured representation of organisational knowledge in the form of ontologies. However, the knowledge asset characterisation is different. In addition, our model extends the REFSENO's formalism by proposing additional knowledge activation components. These are used to capture knowledge asset's history or workers' argumentation that contributed to qualify K-Assets as valid or otherwise.

5. Conclusion and future work

The knowledge model presented in this paper is meant to provide knowledge skeleton for software experience knowledge. It aims to provide ontology-based templates for software experience knowledge. This is in contrast to efforts of software knowledge management systems where less structured knowledge fragments are employed. The proposed knowledge model is intended to represent the backend of any KM tool aimed at facilitating knowledge management activities in a software development organisation. We believe that this model is simple but powerful enough to model various types of software experience fragments. It also incorporates characteristics that support our views about functionalities of intelligent organisational memories (see [5] for details).

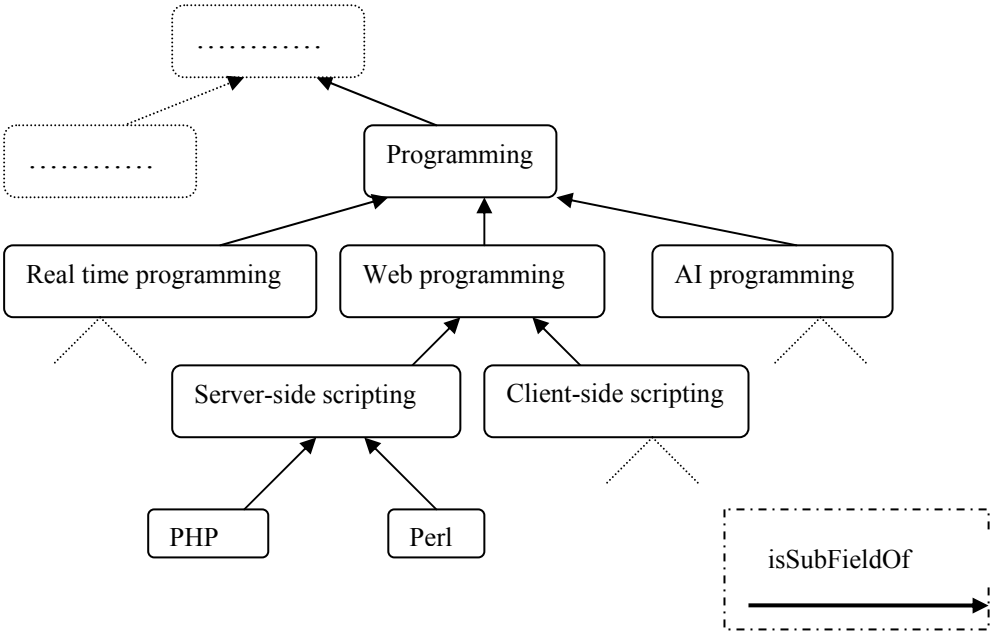

Figure 7: An instance of the competence ontology where generic nodes can be used to retrieve partial searching

We are currently at the last stage of development of a software-oriented KM prototype called LiSER. It supports the framework for collaborative organisation learning proposed in [13]. LiSER's knowledge repository is based on the knowledge model presented in this paper and is implemented as a web-based organisational learning environment. Features and KM activities carried out by the prototype will be published shortly.

References

[1] Nonaka, I, (1998), "The knowledge creating company", *Harvard Business review on Knowledge Managemen*t, Harvard Business School Press.

[2] Vasconcelos, J., Kimble, C. & Gouveia, F. R., (2000), "A design for a Group Memory System using Ontologies". *Proceedings of 5th UKAIS Conference*, University of Wales Institute, Cardiff, McGraw Hill, Forthcoming April.

[3] Brügge, B., Dutoit A. H., (2000), *Object-Oriented Software Engineering: Conquering Complex and Changing Systems*, Prentice Hall, USA.

[4] Conklin, E. J. (2000), "Designing Organisational Memory: Preserving Intellectual Assets in a Knowledge Economy", cited on: 24/5/00 from: http://www.gdss.com/DOM.htm.

[5] Sai, P. L., Mohamed, A. H. and Salim, S. S. (2001), "Towards an Intelligent Organisational Memory System", *Knowledge Management International Conference and Exhibition (KMICE 2001)*, Lankawi, Malaysia,.

[6] Ramesh, B., and Dhar, V. (1992), "Supporting Systems Development by Capturing Deliberations during Requirements Engineering", *IEEE Transactions on Software Engineering*, Vol. 18, No 6, pp. 498-510, June.

[7] Berlin, L.M., *et al*, (1993), "Where did you put it? Issues in the design and use of a group memory", *Proceedings of the INTERCHI'93 Conference on*

*Human Factors in Computer System*s, 23-30, New York: ACM.

[8] Ackerman, M.S. & Malone, T.W. (1990), "Answer Garden: A tool for growing organisational memory". *Proceedings of the Conference on Office Information System*s, 31-39, New York: ACM.

[9] Terveen, LG, Selfridge, PG, and Long, MD, (1999), "Living Design Memory: Framework, System, Memory: Framework, System, and Lessons Learned", *Human-Computer Interaction*, Vol. 10, No.1, pp. 1-37.

[10] Tautz, C.;Wangenheim, C.G.V. (1998), "REFSENO REpresentaion Formalism for Software Engenieing Ontologies", *IESE-Report* NO. o15.98/E, Version 1.1,.

[11] Henninger, S. (1996), "Accelerating the Successful reuse of Problem Solving Knowledge Through the Domain Lifecycle", *Fourth International Conference on Software reuse*, Orlando, FL, pp. 124-133, IEEE Computer Society Press.

[12] Henninger, S. (1998), "An Environment for Reusing Software Process", In *Proc. of the 5th IEEE International Conference on Software Reuse* (ICSR5), Victoria, BC, Canada,.

[13] Mohamed, A. H., Peck, L. S., Salim, S. S. (2002), "A Framework for Collaborative Organisational Learning: A catalyst for continuous software process improvement", in *Proceedings of the First International Conference on Information and Management Sciences*, Xi'an, China, pp. 1-10,.

[14] (Compendium) Internet Web Page URL:http://www.compendiuminstitute.org/tools/questmap.htm