A Framework of Ontology-supported Knowledge Representation in Software Process

Jianying He Haihua Yan Chao Liu Maozhong Jin

School of Computer Science & Engineering, Beihang University, Beijing 100083, P. R. China

Abstract

In this paper, we discuss the crucial importance of properly analyzing and representing useful assets obtained in the process of software development. An ontology supported knowledge representation scheme is presented to compose and organize process knowledge into an organizational repository. According to the characteristics of process knowledge, we build ontologies and utilize them as shared and controlled representation vocabulary that can eliminate conceptual and terminological confusion. In order to facilitate the storage and dissemination of process knowledge, we design a framework named OnSSPKR to validate our proposed ontologies and to confirm the implementation feasibility.

Keywords: Ontology, Knowledge representation, Software process

1. Introduction

The development of software is one of the most knowledge intensive processes that occur in modern organizations and involves many intellectual assets, including developers' personal skills, technical artifacts, best software development practices, and even the whole process experiences. It is necessary to accumulate and disseminate these process assets through the organization. By doing this, developers can facilitate organizational memory to concentrate their creativity on solving technical problems, rather than reinventing the wheel [1]. However, how can these valuable process assets be appropriately and effectively composed and organized into useful asset libraries? This can be generally akin to a problem of knowledge representation which is still one of the main directions pursued by researchers and practitioners.

A number of schemes and programs that are either similarly addressing the need for representation of process knowledge or at least generating interest in the topic, such as ProKnowHow [2], BORE [3], ASPEN [4] and SPO [5]. Although these efforts generally provide methods to capture and register software process related knowledge, there still remains much work to do. Some of the problems emerged in their studies go following:

- **Incompleteness:** Almost all existing studies are based on the concept of Experience Factory [6] that concentrated on formal description or software process models and attempted to capture the whole software process experience as a lesson learned. However, software process knowledge should be systematically analyzed and collected. It should contain both formal and informal knowledge, not only process experiences.
- **Ambiguity of software process knowledge types:** Any type of knowledge that generated and used during software development process should be regarded as software process knowledge [2]. In fact, this process knowledge can be generally divided into the following three types: (1) process experiences; (2) knowledge artifacts; (3) personal skills. However, most of the current research only shed light on the former two types and failed to intentionally consider the third one.
- **Effectiveness of supporting tools:** Since the representation of process knowledge manually is a hard task, offering an automated support for this task becomes an important challenge.

To address the above problems, we tentatively put forward a distinct though related approach, presenting manually constructed ontologies that incorporate all these three types of process knowledge. The ontologies are built according to the characteristics of these three type knowledge and serve as shared and controlled vocabulary that can eliminate conceptual and terminological confusion. Moreover, we have developed a conceptual framework called OnSSPKR (Ontology Supported Software Process Knowledge Representation) which can not only explicitly compose and organize of software process knowledge, but also provide support for project auditors in

existing projects and for project managers in planning new projects.

The rest of paper is organized as follows: Section 2 briefly discusses previous work on software process knowledge representation and related tools; Section 3 presents our ontology supported knowledge representation scheme including detailed analysis of characteristics of software process knowledge and design of corresponding ontologies; Section 4 reports the OnSSPKR framework; Section 5 concludes the paper.

2. Related work

Several works have exploited the use of knowledge representation techniques to support organizing, storing and eventually managing of software process knowledge, such as BORE, ProKnowHow, ASPEN and SPO.

BORE (Building an Organizational Repository of Experiences) is a prototype tool designed to further explore and refine the requirements for tools supporting experience based approaches [3]. As the name indicates, the BORE tool aims reusing organizational experience through packaging it in experience repositories.

In [2], a tool named ProKnowHow is developed to support the standard process tailoring for the projects, allowing the knowledge acquired in this process to be shared. Also, ProKnowHow is based on a standard established software process and could collect and disseminate the knowledge acquired during standard process instantiation.

J. G. Doheny and I. M. Filby propose an ASPEN process modeling framework for modeling and assessing software development processes [4]. Their framework is based on process ontology and can model in an explicit form the contents of software development standards and other forms of best practice.

In [6], an OWL based ontology for software processes, called SPO (Software Process Ontology), is designed and extends to generate ontologies for specific process models, such as CMMI and ISO/IEC 15504. SPO is proposed to support software process definition and assessment. However, this work's concern only relies on process modeling and formal description. It failed to deal with other formal and informal process knowledge.

3. Ontology supported knowledge representation in software process

As mentioned above, software process knowledge can be generally categorized into three types: (1) process experiences; (2) knowledge artifacts; (3) personal skills. To represent and organize these intellectual assets into an organizational repository urges us to give a deep and thorough investigation about their characteristics. By referring to previous related studies, we tentatively proposed a conceptual model of software process knowledge representation (shown in Figure 1). It deals with this three type knowledge differently according to their characteristics. Each type of knowledge has its own ontologies that built manually. Accumulation of the ontologies' instances constitutes an organizational knowledge repository.

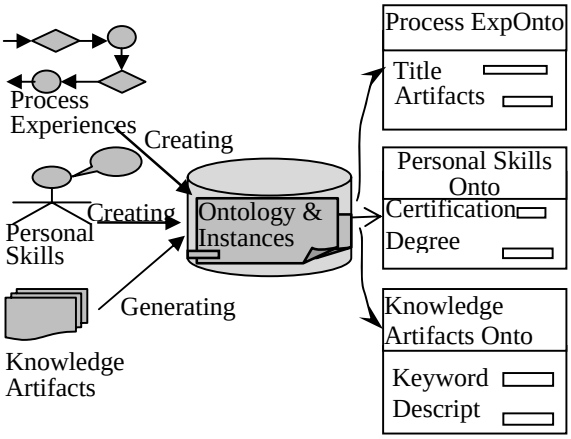

Fig. 1: Conceptual model of software process knowledge representation.

This section discusses how to utilize ontology to formally represent these three types of process knowledge. We also analyze each type of knowledge's characteristics.

3.1. Process experiences ontology

At the core of a software organization's memory, it is process experiences or lessons learned, enabling reuse and sharing of organizational knowledge [2]. It is necessary to disseminate the process experiences through the organization. An organization that does not register the successes or failures of its projects will have as a result the repetition of the failures. However, to compose and organize the process experience tends to be a difficult work because every software development project is unique in some sense. Characteristics of process experiences mainly include:

- **High dependency on specific project:** Application domain, team features, development technology and project size, among other factors influent the way a

software product is developed, operated and maintained [7].

- **Hard to follow a given process model:** None of existing process models can be applied to every kind of software practice.
- **Reusable between similar projects:** Historical best practices can be used for reference in future.

Concerning the above characteristics, we employ ontology technique to formally describe process model and instantiate it to form a tangible software project. Although the existing software process models are different and their model components have various names, they nevertheless have some similarities (see table 1), e.g. their main components are "Process" and "Practice" [6] [8] [9]. Normally, the models have a set of processes, which could guide the software production, and the processes are classified into several domains, called "Subsystem" or "Category".

Components / Model	Sub system	Catg	Proc	Sub Proc	Practice	Proc Attr
CMM		Catg	Key Proc Area		Key Practice	
CMMI		Catg	Proc Area	Specific Goal	Specific Practice	Generic goal
ISO/IEC 15504		Catg	Proc	Component Proc	Basic Practice	Proc Attr
ISO9001	Sub system		Main Topic Area		Management Issue	
BOOTSTRAP	Proc Area	Proc Catg	Proc		Practice	

Abbreviation: Proc=Process; Catg=Category; Attr=Attribute

Table 1: Similarities of existing standard software process models [5].

With the capability of OWL, we design a process experiences ontology (RDF Graph is shown in figure2) based on the distilled process model presented in [5]. The proposed ontology defines the process model at the schema level and plays a similar role to conceptual data schemas in the database community. Thus a project's software process definition is in fact an instantiation of the standard process.

In this ontology, we define some core classes to represent components in models, and properties to represent the relationship between components. And the core classes are chosen from the shared concept in CMM, CMMI, ISO/IEC15504, ISO9001 and BOOTSTRAP [10]. Then we group them into three conceptual sets, which become the major components in our tree hierarchy. Finally, we develop the full hierarchy tree inserting all the terms we have collected previously into our Classes using a kind of relationship.

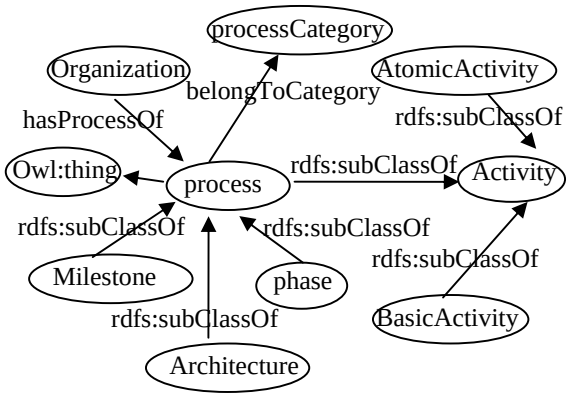

Fig. 2: Process experiences ontology.

3.2. Personal skills ontology

Developer's skill involved is the most important knowledge needed to capture and represent because all activities are carried out by human in the final analysis. Developers' personal skills directly determine the quality of the process through software is developed. So, knowledge in developers' mind has to be systematically collected, stored in a corporate memory, and shared across the organization. However, to do so is a rather challenging task. Characteristics of personal skills include:

- **Tacit:** developers' skill will remain latent in their mindset and never become explicit without being elicited.
- **High value:** unquestionable, employees' tacit knowledge is one of the organization's most valuable intellectual assets.
- **Unstable:** knowledge in people's mind tends to leak when individuals leave the company.

Our personal skills ontology is defined in a hierarchal structure of different kinds of skills a person can possess regarding the above characteristics. Skills are obtained from various sources, such as certified training, formal education or previous work experience [11]. Because skills have a great effect on the way employees carry out tasks, the assignment of tasks to proper employees is crucial to the potential success of a project.

3.3. Knowledge artifacts ontology

An organization's knowledge repository should also contain the knowledge artifacts obtained throughout

the projects. The knowledge artifacts could be a list of requirements, a case tool graphic or some software code. Generally speaking, there are four major artifact types: technical, quality, safety and management [12]. Moreover, they can be subcategorised, for example, quality artifacts have verification and validation subtypes.

Although there is a growing interest in tools and methods that support ontology automatic generation from document corpus, output results are far from mature and practical [13]. Therefore, we propose an ontology indexed document corpus approach to represent knowledge artifacts (see in figure 3).

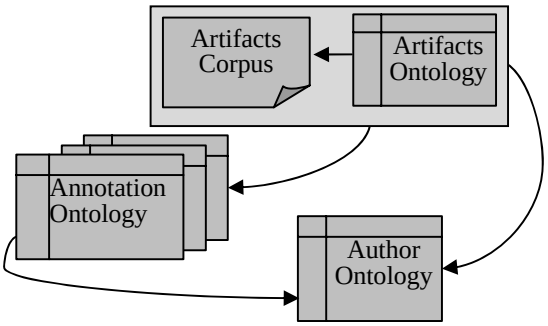

Fig. 3: Ontology indexed document corpus

The figure shows that main body of explicit knowledge still lies in "Artifacts Corpus". However, an artifacts ontology is designed and logically bound to this "Artifacts Corpus" which has explicit representations for the information and important knowledge concepts contained in artifacts.

Every type of document's ontology can be designed respectively. But it usually has the sections like "Title", "Created Date", "Rationale", "Dependencies", and "Version" etc. An exceptional case is the section "Annotation" which is a set of ontology too. It carries document receptor's annotating information. This occurs similarly in the case of "Author" section which is public ontology describing document creator's basic information.

4. OnSSPKR knowledge representation framework

Under the ontology groundwork, we design a prototype named OnSSPKR framework to prove our idea. The OnSSPKR framework is currently implemented using J2EE/EJB technology and Browser/Server based. It mainly provides with K-Manager Environment and K-User Environment. A diagram of its architecture is shown in Figure 4. The following sections describe the OnSSPKR framework in more detail.

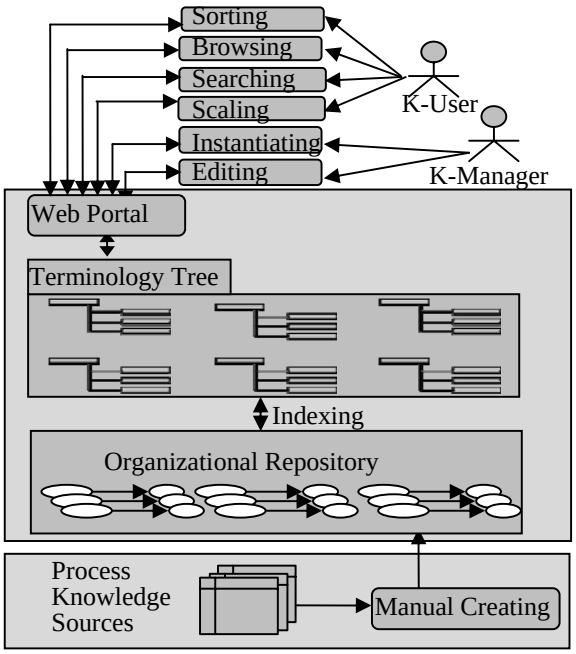

Fig. 4: The OnSSPKR architecture.

4.1. Design goals

OnSSPKR was developed to achieve the following goals:

- To support useful process assets to be safely composed and organized into an knowledge repository;
- To represent the knowledge generated and acquired during process of software development;
- To support retrieval and dissemination of stored knowledge from organizational repository.

To satisfy the above requirements, OnSSPKR is designed to have the architecture shown in figure 4.

The process knowledge sources contains both formal and informal knowledge varied from quality models, development artifacts, experiences, lessons learned to personal expertise. As pointed above, it is divided into three types: (1) process experiences; (2) knowledge artifacts; (3) personal skills.

The organizational repository stores ontology and its instances we discussed in section 3. It is also divided into three: (1) process experiences repository; (2) knowledge artifacts repository; (3) personal skills repository. Moreover, the organizational repository is indexed and organized into a terminology tree derived from the major concepts defined in system ontologies.

On the top layer of OnSSPKR are functions provided for project managers and common users,

called K-Managers Environment and K-User Environment respectively. Next, we discuss them in detail.

4.2. K-Users environment

Common users in an organization(called K-Users) participate in organization's development activities and access their knowledge through either their own knowledge sources or knowledge sources obtained from outside.

OnSSPKR offers K-Users functions to sorting a concept from the terminology tree and register their knowledge in an indexed folder. Also, K-Users can search or browse the knowledge repository to find relevant knowledge for their jobs. In addition, after assimilating the knowledge they acquired, K-Users can scale on the items they viewed as a feedbacks to the repository. Detailed description of K-Users environment's functions go following:

- **Sorting:** select a concept from the terminology tree and store knowledge in the indexed folder. This is a process of knowledge composing and storing.
- **Browsing:** brow the whole terminology tree to have an overall vision of the organization's intellectual assets.
- **Searching:** search and reference knowledge items based on terms. This is a process of knowledge disseminating.
- **Scaling:** track feedbacks to the knowledge viewed. This is a process of lesson learning.

4.3. K-Managers environment

Knowledge Managers (called K-Managers) lead organization learning by editing ontologies and instantiating a software process for a project. K-Managers are also responsible for adapting and approving process knowledge items input by K-Users. Therefore, all functions in a K-Users environment will be informed to K-Managers. This information will give K-Managers more opportunities to collect new concepts of K-Users.

OnSSPKR offers two additional functions for K-Managers described as follow:

- **Editing:** create and revise ontologies together with their instances to build the foundation of the system.
- **Instantiating:** instantiate a software process for a new project.

5. Conclusion and future work

We have presented an ontology supported knowledge representation approach for process assets. We generally divided the process knowledge into three types, i.e., process experiences, knowledge artifacts and the personal skills, and discussed their characteristics respectively. Ontologies towards these types of knowledge were also built to serve as shared and controlled vocabulary. Finally, a framework named OnSSPKR was designed in this paper as an embodiment of this idea.

Key benefits and potential usage of our work include:

- Supports to compose and organize useful process assets into a knowledge repository;
- Supports to accumulate software process knowledge for further retrieval;
- Potential use of facilitating project's feedback to make software process improvement easier.

It should be noticed that this study has examined only knowledge representation issues in software process. It is not a knowledge management scheme. However, we have employed OnSSPKR to compose and organize our organizational intellectual assets and the output results have demonstrated that the three knowledge types can cover almost all the useful process assets and our framework can effectively retrieve stored knowledge.

Our next step work involves consummating our designed ontologies, infiltrating knowledge categorizing algorithm into the framework and extending the functions of the tool.

Acknowledgement

This work is partially supported by National Natural Science Foundation of China (Grant No.60573084) and National Weaponry & Equipment Foundation (Grant No. 9140A15050106HK0114).

References

[1] A. Birk, D. Surmann and K. Althoff, Applications of knowledge acquisition in experimental software engineering. *Proc. of the 11th European Workshop on Knowledge Acquisition, Modeling, and Management*, pp. 67-84, 1999.

[2] Ligia da Motta Silveira Borges and Ricardo de Almeida Falbo, Managing software process knowledge. *Proc. of the International Conference on Computer Science, Software Engineering, Information Technology, e-Business, and Applications*, Foz do Iguazu, Brazil, June, 2002.

[3] Scott R. Henninger, Tool support for experience based software development methodologies. *Advances in Computers*, 59:29-82, 2003.

[4] ww.aiai.ed.ac.uk/project/ftp/documents/1996/96-ausda_tr_escom.doc.

[5] Li Liao, Yuzhong Qu and Hareton K. N. Leung, A software process ontology and its application. *Proc. of the 4th International Semantic Web Conference*, Galway, Ireland, November, 2005.

[6] V. Basili, G. Caldiera and H. Rombach, *Encyclopedia of Software Engineering*, John Wiley & Sons, New York, 1994.

[7] W. Scacchi, Understanding software process redesign using modeling, analysis and simulation. *Software Process Improvement and Practice*, 5:183-195, 2000.

[8] M.G. Mendonça Neto, V. Basili, C.B. Seaman and Y-M Kim, A prototype experience management system for a software consulting organization. *Proc. of the 13th Int. Conference on Software Engineering and Knowledge Engineering*, Buenos Aires, Argentina, 2001.

[9] S. Stab, R. Studer, H.P. Schnurr and Y. Sure, Knowledge processes and ontologies. *IEEE Intelligent Systems*, 16:26-34, 2001.

[10] M.C. Paulk, C. V. Weber, B. Curtis and M.B. Chrissis, Key practices of the capability maturity modelSM. *Technical Report*, CMU/SEI-93-TR-025, SEI Carnegie Mellon University, 1993.

[11] Philip Nour, Ontology-based retrieval of software engineering experiences. *Master Thesis*, Department of Computer Science, University of Galary, 2003.

[12] P. Ceravolo, E. Damiani, M. Marchesi, S. Pinna, and F. Zavatarelli, A ontology-based process modeling for XP. *Proc. of the Tenth Asia-Pacific Software Engineering Conference*, pp. 236-242, 2003.

[13] A. Abecker, A. Bernardi, K. Hinkelmann, O. Kuhn and M. Sintek, Toward technology for organizational memories. *IEEE Intelligent systems*, 13: 40-48, 1998.