Role-Activity-based Knowledge Management System for Software Development Processes

Saw Sanda Aye
Ochimizu Lab
sawsanda@jaist.ac.jp

1. The purpose of research

Most of knowledge on software development becomes old-fashioned in several years because the technologies are rapidly advancing. We always need new relative knowledge to improve our skills and products to adapt with new paradigms and new technologies. However, it is difficult to get all or some of the conceptually relative knowledge when we desire to pick up for our purpose.

Ontology can provide basis for systemizing knowledge and manifesting the tacit knowledge. By taking the advantage of ontological concept, we try to support knowledge to development process. Based on roles activities in development process, we define activity patterns. And combine these patterns to actual development process to improve reality and effectiveness of processes.

We also try to improve on communication path between developers by combining process with organization patterns based on role-activity model to reduce communication overhead in development process.

2. Progress has been done

2.1. An Ontology for Complexity Issues in Software Engineering

It is difficult for developers or maintainers to get proper knowledge efficiently and effectively when they encounter the problems during their software engineering activities because knowledge on software engineering has a wide spectrum and there are many similar concepts that have different usages. To deal with this problem, IEEE and ACM published SWEBOK (Software Engineering Body of Knowledge) to provide knowledge necessary for developing and maintaining software. The knowledge presentation structure of SWEBOK, however, has the problem. It is mainly organized based on phasing structure of a waterfall model. So there remains the problem that it is very difficult to understand the relation of knowledge through different phases. By taking the advantage of ontological concept, we try to support knowledge to development process based on roles' activities.

We try to re-organize the knowledge on software engineering based on the following policy.

1. Arrange the concepts in the problem domain that developers or maintainers may encounter during their software engineering activities.
2. Arrange the concepts in the solution domain that researchers have invented to deal with the problems.
3. Arrange the concepts in the techniques domain that should be adopted to solve the problems.

We take an ontological approach to organize the knowledge structure. An ontology is a system of concepts. We scope on the complexity issues to define an ontology. We follow the next two steps to define the ontology.

1. Define common concepts in a domain by arranging the concepts described in publications like papers and books.
2. Define the relationships among those concepts to define an ontology.

The most important part of the ontology design is the relationship design among those concepts because they are the key information that enables us to access the proper knowledge. For example, we adopt the information hiding principle to localize the impact of change on data structures. So, “localize” is a one of the key relationship to organize the ontology.

We adopt the concepts in problem domain mainly from F. Brooks’ work, the concepts in solution domain from principles in software engineering developed by researchers in SE field, and the concepts in technology domain from several substantial papers related to object oriented technology, aspect oriented technology and product line engineering technology.

We also define the interfacial structure to access the knowledge defined in the ontology, our interfacial structure is defined as several activity templates like designing, programming, or testing.

Starting from these activity templates, if someone encounters the problem related to complexity issues, they can access the concepts in the problem domain. By expressing how they want to solve the problem with help of the concepts in solution domain, they can finally access the proper techniques that help them to solve the original problem.

2.2. A Process Model

In the development process, besides producing artifacts, there should be proper communication support since the communication is fundamental to organizational success. Everyone involved in the project must be prepared to send and receive communications, and must understand how communications in which they are involved as individual affect the project as whole. Although there are many attempts to support communication, to acknowledge the communication aspect in software development process was inattentive. In order to take into account of project communication management aspect in the process, we propose a process model that combines the RUP (Rational Unified Process) with Organizational Pattern based on role-activity model.

We define meta-model of roles by investigating the RUP, identify the shared artifacts, and the developers who share the artifacts. The relationships among roles are defined by means of relationships of the shared artifacts created by each role and these relationships are illustrated as role management model. Then we combine RUP with J. Coplien's organizational patterns by using role management model. Each role of an organization pattern and each role of RUP are compared. If both of them have similar activities, they will be combined as shown in Fig 1. This model points out the communication path which should be taken and which should not be taken, for instance, the project manager shields the development personnel from interaction with external actors by pattern 24 fire walls. This causes communication overhead to decrease and the participants to clarify the process. So, we can improve the quality, productivity, efficiency and effectiveness of the development process.

3. Future Direction

We are going to validate our process model with mathematical approach to know what the effective way of communication of this process model is. We have already developed a model and tool that can thread for each topic to represent the streams of messages in the communication link for e-mail communications. This information is extracted to the information repository. We intend to analyze our process model with actual messages that passed through a system by using the information repository.

We defined the ontology for complexity issues in software engineering but it still needs to refine to cover all the concepts in software engineering. We try to represent our ontology in OWL. And we are also developing a prototype to show the usage of the ontology.

4. List of publication

1. "Process Model Combining the Artifact Centered Process with Communication Path", Saw Sanda Aye, Yi Zhou, Koichiro Ochimizu, The 5th International Workshop on Software Process Simulation and Modeling (ProSim'2004), p 13-21.
2. "Defining Ontology for Complexity Issues in Software Engineering", Saw Sanda Aye, Mitsuru Ikeda, Koichiro Ochimizu, Conference on Japan Society for Software Science and Technology, 2004.