Modeling Software Processes by Using Process and Object Ontologies

Takahira Yamaguchi
School of Information, Shizuoka University
3-5-1 Johoku Hamamatsu, 432 JAPAN
yamaguti@cs.inf.shizuoka.ac.jp

Abstract

In order to model software processes based on ontologies engineering techniques, this paper presents a methodology to manually construct the following ontologies: an object ontology based on constituent elements for objects, and a process ontology based on relationships between inputs and outputs, such as subsumption relationships. Next, using the constructed ontologies, software process plans are generated for user queries, with both user interaction and constraint satisfaction by generate and test paradigm. Furthermore, experimental results show that the methodology works well in generating software process plans good for a query about a software process plan from a basic design and to a detailed design.

1. Introduction

Recently, several research projects have focused on integrating the following two fields: software process engineering and knowledge engineering. In particular, a USC group has developed a knowledge-based environment for software engineering processes, called Articulator [P.Mi90]. The environment modifies software process plans with faults, by diagnosing, replanning and rescheduling them, using several types of knowledge and strategies. This paper focuses on how to integrate the following two fields: software process engineering and ontologies engineering.

2. Constructing Ontologies

An ontology provides a specification of a conceptualization. This section focuses on domain and process ontologies, providing a methodology for manually constructing and connecting them.

After gathering and analyzing many general objects and real-time specific objects in software processes, I found that the objects have been composed of the following four or less constituents: components for computer systems, attributes for the components, operations for the components and information media to represent information of components, attributes and operations. In developing an object hierarchy, these four nodes are placed as sub-nodes of the root node 'object' and more detailed objects have been branched down from the four sub-nodes. So far an object hierarchy includes more than four hundreds objects.

After gathering and analyzing many software processes as well as objects, I put the following top-level processes based on subsumption relationships between inputs and outputs: 'Analyze' process (inputs subsume outputs) and 'Synthesize' process (inputs do not subsume outputs). 'Analyze' branch has out into 'Evaluate' process (outputs are subsumed inputs after evaluation) and 'Extract' process (outputs are subsumed by inputs without evaluation). 'Synthesize' branches out into 'Make' process (all outputs are generated) and 'Modify' process (some outputs are generated and others not).

Since specified processes have specific properties, the above-mentioned branching does not work at lower-level processes. Therefore, I have identified a process scheme as a model to represent software engineering processes. The schema's roles have been divided into three categories: information resources (input, scope in input, output and reference), related processes (predecessor process and successor process) and other resources (agent and tools). So far a process hierarchy with about two hundred and fifty processes have been developed based on this method.

By filling the objects in the input and output roles in the process schemata, the object and process ontologies have been connected. This connection is helpful for finding a best software engineering process plan for user queries, as explained later.
3. Engineering Softw are Processes Using Ontologies

A process-centered software engineering environment can be designed using the process and object ontologies.

STEP 1: A user asks the environment how to follow the software engineering process (SEP) from one (starting) object to the other (goal) object.

STEP 2: GENERATION subsystem retrieves some processes from the process ontology, which have the starting (goal) object in the input (output) role. When not finding such processes, it is tested if starting (goal) object are included in all subnodes of the input (output) role. Thus both first and last processes for SEP come from the process ontology.

STEP 3: The above-mentioned bi-directional (forward from starting objects backward from goal object) search repeats until joined. When there are still many SEP candidates, the environment asks for more constraints to reduce SEP candidates.

STEP 4: After getting SEP candidates, they are tested against the criteria of SEP specific feature, such as development terms, conflict among resources used for processes, process reliability, with a user interaction. Thus one or more proper SEP candidates remain after the test. When there are still many SEP candidates, the environment asks for more criteria to further reduce SEP candidates.

4. Experimental Results and Discussion

Suppose that a user asks the environment how a SEP is going from the input of 'Requirement Specification of S/W subsystems' to the output of 'Detailed Design of Each Component of S/W subsystems'.

GENERATION subsystem retrieves some processes from the process ontology, which have 'Requirement Specification of S/W subsystems' in the input role. Then three processes, such as 'Design Basic Components of S/W subsystems', 'Design Basic Components of I/F subsystems' and 'Design Basic (logical) Aspects of DB', have been retrieved through a forward search. The subsystem also retrieves other processes which have 'Detailed Design of Each Component of S/W subsystems' in the output role. Then three other processes, such as 'Design Detailed Components of S/W subsystems', 'Evaluate All Detailed Designs', and 'Review All Detailed Designs Together', have been retrieved through a backward search. After repeating this bi-directional search, many SEP candidates have come up. By asking a user for more concrete constraints, such as that it is not necessary to review designs together with a user, the number of SEP candidates will be reduced.

The candidates have been tested against typical SEP specific feature such as development terms, but have not been reduced at all. By asking a user for more criteria, such as it is necessary to evaluate design results after getting them, SEPs have been removed, including a sequence from 'Design Basic Components of S/W subsystem' to 'Detailed Design of Each Component of S/W subsystems'.

When we develop SEPs specific to some application domain using the document, it is important to specialize general SEPs and to identify and remove SEPs unnecessary to the application domain. In this experiment, such specific SEPs have been generated using constraints and criteria from a user. Thus the environment can support a user in developing SEPs specific to application domains.

5. Summary and Future Work

After implementing two ontologies and an environment through the analysis of real software engineering process, it turns out that the environment can generate software engineering processes specific to application domains with user interaction. Based on the work here, I will be planning both to upscale the ontologies and to undertake more practical experiments. We will also take the facilities to maintain large scale ontologies into account.

References