Web-based Process Management System with Object-Centered Process Modeling

Makoto Matsushita[†]
matusita@ics.es.osaka-u.ac.jp

Hajimu Iida[‡]
iida@itc.aist-nara.ac.jp

Katsuro Inoue[†]
inoue@ics.es.osaka-u.ac.jp

[†]Graduate School of Engineering Science
Osaka University
1-3 Machikaneyama, Toyonaka
Osaka 560-8531 JAPAN

[‡]Information Technology Center
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma
Nara 630-0101 JAPAN

Abstract

Most of process-centered software engineering environments and those description languages tend to focus on the way of producing the software product; they describe mainly how software is developed. However, recent software development such as distributed and networked development tends to focus on the software product itself; we are interested in software component, development tool, and so on. These are because of emergence of various types of software development approaches, e.g., object-oriented software development, software re-use, and so on. To fill up the gap between the current process-centered software engineering environments and the emerging software development, we discuss a new process modeling method for describing the emerging software development method, and design a web-based software development environment based on the model. Our model consists of a set of objects, which represent the artifacts and the resources in the real software development environment. We have developed a prototype system of the web-based environments, and we have enacted ISPW-6 software process example with this system. Our system can illustrate software development environment very naturally, and provides an environment for software process execution, management, and improvement.

1. Introduction

Software process description and its enaction help the software development to proceed effectively and to produce high quality software[6, 7, 11]. However, most of process-centered software engineering environments tend to enforce specific types of development activities to the developer. Also, they require proprietary and exclusive systems/environments which are completely different from existing software development environment [2, 3, 4, 5, 12, 13, 14]. Therefore those systems are not yet widely used in real software development.

Most of these software process languages[1] focus on the description of "how to produce a product"; i.e., it describes a process-orinted activity of software development. However, recent software development methods such as object-oriented programming, software reuse, component-based programming mainly focus on "what should be made"; i.e., it does a product-oriented activity to artifacts in software development environment. In process-centered software engineering environment, these artifacts-centered idea of software development should be supported, to make more effective support for software development.

In this paper, we discuss a basic idea of object-centered process description model. We also present the design and the prototype implementation of web-based process management system based on our model, which contains essential features to represent and enact ISPW-6 (International Software Process Workshop) example problem[8]. The goal of the system is to illustrate software development environment as it is, and provide a framework for software description, management, and improvement.

Our model consists of a set of objects, which show the artifacts and the resources in the software development. An object consists of attributes and methods. An attribute represents characteristic of the object. A method is a function applied to the objects. The model provides object inheritance to share information between objects. Messages to objects, which would activate the methods or attribute accesses are recorded automatically as the operation history of the object. With these features, the status of the software development environment is easily monitored, and thus the process management system provides helpful information to the project manager. Our prototype system assumes that it runs under network-based software development environment, and is also used by the developers to help their work.

This paper is organized as follows. In section 2, we will describe the definitions and features of the model, and we compare our model to other software process description languages. In section 3, we will explain an idea of our system, add-on type process management system. Finally, we conclude our work in section 5.

2. Object-Centered Process Modeling

In this section, we introduce our basic idea of object-centered process modeling technique. First of all, the definition of the model is presented. Then we will show how the software process is presented with this model.

2.1 Definition

All artifacts and resources in the software development environment are shown as *Objects*. Software development environment is defined as a set of objects. Object O is defined as (Ol, A, M), where Ol is an object label, A is a set of attributes, and M is a set of methods. Ol is the unique name of this object, used to specify the object.

An attribute $a \in A$ is defined as (Al, Av), where Al is an attribute label and Av is an attribute value. An attribute label is the unique name for attribute and it indicates what kind of information is needed to the object. Information itself is represented by the attribute value. The type of the attribute values is a number, string, label, or list of these types.

A method $m \in M$ is defined as (Ml, Mv), where Ml is a method label, and Mv is a method function. The method label is the unique name for this method function, showing what operation is done with this method. Actual operation is defined by the method function, and defined as a unique mapping among sets of objects (iff the two method label is the same one, these methods are actually the same one).

Figure 1 is an example object description of this model. This description shows an object of a design document.

In this example, four attributes (`owner`, `doctype`, `input`, and `location`) are defined, to show information about this object. For example, an attribute `owner` shows who is the responsible person of this document, and is defined as the other object which represents the person. There are three methods, `modify` for editing this document, `view` for viewing this document, and `review` for notifying the owner to do the review activity.

```
Object Design
      Attribute owner matusita
      Attribute doctype "Design Document"
      Attribute input Specification,Schedule
      Attribute location "/path/of/document"
      Attribute filename "design.doc"
      Method modify
            var editor = geteditor;
            var viewer = getviewer;
            if (viewer)
                  exec(viewer, input);
            if (editor) {
                  viewinput;
                  exec(editor, location, filename);
            }
      endMethod
      Method view
            var viewer = getviewer;
            if (viewer)
                  exec(viewer, location, filename);
      Method review
            notify(owner, "Please review the design.");
      endMethod
endObject
```

Figure 1. Object sample

2.2 Features

The model has various features to support project management. In this section, we show these features with some examples.

Operation History: An operation to an object such as referring attributes is processed by sending a message to the object and activating a method of the object. In this model, any operation to all objects are recorded as a history[1]. Operation history is also stored in an special attribute. The attribute history records a list of labels of the object which operates the attribute, operation time, and the contents of referenced value/changed value. The method history records a list of labels of the object which executes the method, beginning and ending time of the execution, and the results of execution.

These histories are automatically recorded, i.e., reading an attribute is really achieved after recording to the history, and executing method is really achieved after recording to the history.

[1]This is possible under our assumption of the granularity of the model description. The descriptions are generally not so in fine granularity or they do not create and delete objects so intensively as scientific calculation.

Mapping Between a Model and Project Environment: We need mapping between described process model and the project software development environment. The mapping is achieved by our system; a software development management environment based on the model. User interface of the system is provided to manage objects directly. With this interface, the developers can operate objects easily. Periodical search mechanism to files is used to synchronize automatically between the model to the project environment. While a method is executed, files are modified/referenced or tools are invoked. The changes of the status in the model influence to the project environment.

Object creation: Object creation is achieved by inheriting attributes and methods from `.OBJECT`, a predefined object template, or from any existing object. Any object creation is based on this object inheritance, therefore following explanation is focused on object inheritance.

Assume object `X` is already defined, and object `Y` inherits from `X`. First, `Y` is created as a *copy* of `X` but it does not have any attributes/methods which are not allowed to be inherited. Then, a modification and addition to `Y` is processed. In this model, inheritance is accomplished with a copy of existing object associated with newer definitions and additional definitions for the new object.

Object inheritance is triggered by executing a predefined fork method, which is already defined in template object. This method gets parameters which are used to modify self-copied object. Redefine the fork method allows to define object-specific initialization.

3. Web-based Process Management System

Figure 2 illustrates our process management system. This system works on network-based environment, and co-exists with existing environment. It uses an open web technology, and it is designed for both software developers and project managers. It is composed of server system and client system.

The server system consists of the repository, repository programs, the method engine, the method sender, the HTML (HyperText Markup Language) translator, and the web server. The server system manages the objects described with a model, executes a method, and generate an user-interface system for the client system. The client system consists of a web browser and the method executor.

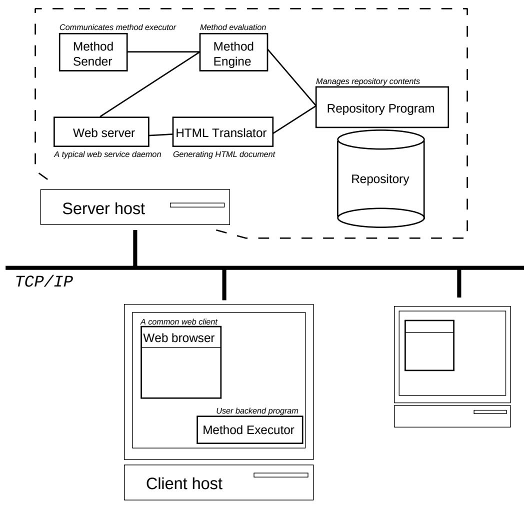

Figure 2. System Overview

3.1 Server System

There are mainly three parts in the server system: the repository part, the method execution part, and user-interface part. All of the components are running in a server host.

Repository Part: The repository is for a object storage management. It handles the object descriptions, and collects/stores the information in software development environment such as the result of method execution and so on. Actually, the repository is files and directories of the filesystem: objects are mapped to the directories, and attributes/methods are mapped to the files.

The operation of the repository is achieved by the repository programs. Each operation has a repository program. There are seven types of repository programs: list the attribute, modify an attribute value, show an attribute value, show the object location, list the methods, show the method contents, and list the object itself.

Method Execution Part: The method engine part is for the interpreter of method description. It works as the core part of process execution. It is also communicates with other parts (repository and user-interface part), and manages the behavior of the whole system.

The method sender is a sub-component of method engine. It manages client-side execution by method executor; it sends a job to method executor, and waits until the job is finished, and the results is returned to method executor. Operations to the attributes (read/write a value), the contents evaluated method,

and the returned results from method executor are used as the basic data for process management.
User-Interface Part: The user-interface part is for interactions between the system and the users. It provides the feature to show the information of the objects, to enact process with a method, to provide the information of the result of past activity, and so on. This part consists of the HTML translator, a web server/browser, and the method executor. The HTML translator translates the repository information from/to the HTML format, i.e., we employ a Web system for the system of human-computer interaction. The web server is usually used in any place. We employ the Apache http server for the system, and modifies to communicate the other components of the system each other. The method executor receives the actual method contents from the method sender, evaluates the method, and returns the execution result.

3.2 Client System

A web browser is also commonly used in the world. We assume that the most development environment have already installed such browser. The method executor is for user-side execution. The execution contents is passed from method sender, and the method executor executes the contents (also invokes other programs if needed), and returns a result of the execution. Each user runs a web browser and one method executor for a client system.

3.3 Prototype Implementation

We implement this system on FreeBSD operating system with PentiumPro-based Personal Computer. We used the Perl language to construct this system to establish a high portability of the system. However, we do not hesitate to use C language for the mission-critical components such as method execution. Our system also employs the components commonly used in many places. This is because these components are robust enough, and cause low impact to existing development environment.

4. Case Study: ISPW-6 Example Problem

In this section, we show a case study of our system, an application to ISPW-6 Example Problem. First, we describe what is the ISPW-6 example, then we show the way of an application of the system to the problem.

4.1 ISPW-6 Process Modeling Example

As a software development process to model, we employ ISPW-6 Example Problem[8], which is proposed by Kellner et el in the 6th International Software Process Workshop. This example is commonly used to provide the basis of comparison and evaluation of various software process modeling and the execution system. There are also several extensions to the example, however, we only uses the core example problem.

The core example problem specifies a process of modifying single module in a system. This process is composed of eight sub-steps: there are specifies scheduling task, design task, review task, coding task, test task, and monitoring task. Each sub-steps are described as input and output artifacts, activity and its requirements, engineers' role and organization. These tasks, artifacts and its relationship are shown in figure 3.

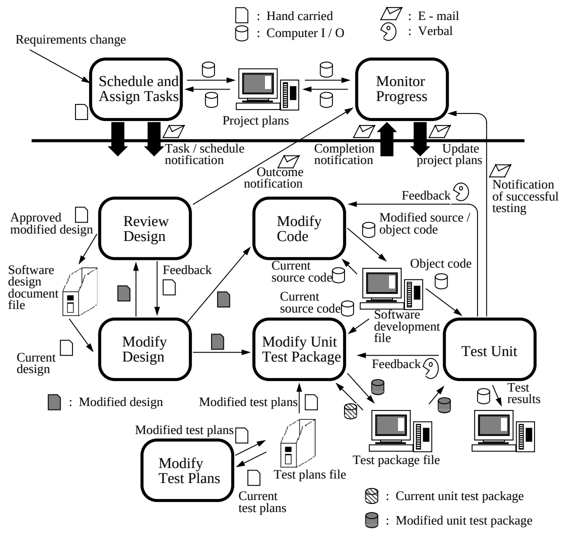

Figure 3. ISPW6 Process Modeling Example

4.2 Application of the System

In order to confirm that our prototype system is functionally valuable, we applied ISPW-6 process modeling example to the system.
Analyze the Example: At first, we should decide the real situation of software development environment with ISPW-6, since the example only specifies the role of the engineers and the type of artifacts; nothing are described about the number of engineers, document filename, and so on.

In this application, we assume six engineers; one project manager, two design engineers, two quality assurance engineer, and one member of configuration control board. They are teamed up, and work on the same network. For the simplicity, we assume that there are only one file for each type of artifacts.

Describe as Objects: We found that there are three types of objects in the example; artifacts which are used in the sub-steps, engineers, and the role of engineers. As a result, we extracted from the example and describe thirteen objects for artifacts, six objects for engineers, and four objects for engineers' roles. Figure 1 shows the description of the design document.

The artifacts objects are used for describing the type, location, applied activity, and so on. The engineers objects are used for describing the name of engineer, favorite tools and personality. The engineer's roles object is used for describing the role-oriented activity for the engineer.

Enact the System with Described Objects: Finally, we import the objects to the system and enact the system. Following is the sample case of system enaction.

When a developer execute a method, first, he/she use his/her own web browser to see an object which has a method to execute. Note that all objects and its attributes/methods are mapped into URL (Uniform Resource Locator), web server and HTML translator translates from the URL to object contents in HTML document. Then, the developer click the anchor of the method with a mouse. This action is passed to the web server, and the method engine are kicked for processing this method. The method engine knows who invokes the method, and evaluate the description of the method contents. The method engine communicates the method sender if some commands need to be executed. The method sender sends a command to the method executor, then the method executor invokes a command within his/her development environment.

Figure 4 is a screen-shot of the system. This is a typical UNIX-based development environment, and web browser shows the contents of object. Method invocation can be done with the click of the mouse; in this screen-shot, "coding" method of source code object is clicked and the editor comes up with source file.

With this test-drive of the prototype, we have confirmed that the process description is enacted and the resulting environment established a proper behavior of the ISPW-6 example. We also found that the prototype system provides the feature of software process management; current status of each objects, the result of past activities, and so on.

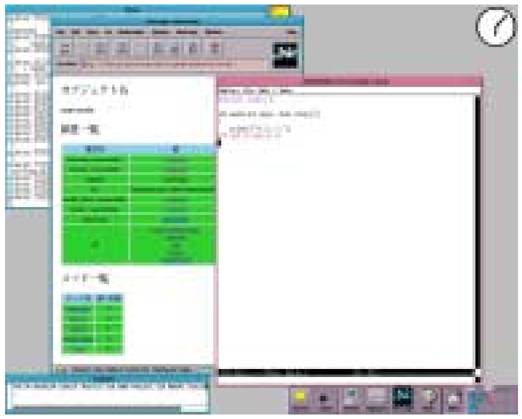

Figure 4. Screen-shot of Prototype System

5. Conclusion

In this paper, we discuss a basic idea of object-centered process description model and its web-based implementation to support software development management.

Our model consists of a set of *Objects*, to represent all artifacts in software development environment. With this model, software development environment itself is illustrated as it is; that is powerful capability of the process management.

Our system is a web-based software development environment. The system can be used with existing development environment, and it provides product management and reactive activity execution support. A prototype has been implemented and ISPW-6 example has been described and executed on the prototype.

For a further research, more sophisticated process model to describe the "situation" of ongoing software development is desired. We have already designed MonoProcess process modeling approach[9] based on the model described in this paper, and design the enhanced system especially for process management[10]; however, the idea of object-oriented architecture such as object reflection model should be employed. We have already tried a full implementation of the system. Validation of our model and more support for process enaction are also planned.

Acknowledgement

We are grateful to Tetsuo Yamamoto and Yutaka Fujiwara for their contribution of the development of this prototype.

References

[1] P. Armenise, S. Bandinelli, C. Ghezzi, and A. Morzenti. Software Process Representation Language: Survey and Assessment. In *Proceedings of the 4th Conference of Software Engineering and Knowledge Engineering*, pages 455–462, 1992.

[2] S. Bandinalli, E. Nitto, and A. Fuggetta. Supporting Cooperation in the SPADE-1 Enviornment. *IEEE Transaction on Software Engineering*, 22(12):841–865, 1996.

[3] S. Bandinelli, A. Fuggetta, and C. Ghezzi. Software Process Model Evolution in the SPADE Environment. *IEEE Transactions on Software Engineering*, 19(12):1128–1144, 1993.

[4] S. Bandinelli, A. Fuggetta, and S. Grigolli. Process Modeling in-the-large with SLANG. In *Proceedings of the Second International Conference on the Software Process*, pages 75–83, 1993.

[5] I. Ben-shaul and G. Kaiser. A Paradigm for Decentralized Process Modeling and its Realization in the Oz Environment. In *Proceedings of 16th International Conference on Software Engineering*, pages 179–188, 1994.

[6] K. Inoue. Current Research Activities on Software Process. *Japan Society for Software Science and Technology Technical Re port*, 95(SP-2-1):1–10, 1995.

[7] T. Katayama. Software Processes and Their Research Topics. In *11th Conference Proceedings Japan Society for Software Science and Technology*, pages 433–436, 1994.

[8] M. Kellner, P. Feiler, A. Finkelstein, T. Katayama, L. Osterweil, M. Penedo, and H. Rombach. Software Process Modeling Example Problem. In *Proceedings of the 6th International Software Process Workshop*, pages 19–29, 1991.

[9] M. Matsushita, M. Oshita, H. Iida, and K. Inoue. MonoProcess: Software Process Description Language with Object Model. *Information Processing Sciety of Japan Technical Report*, PRO-14:97–102, 1997.

[10] M. Matsushita, M. Oshita, H. Iida, and K. Inoue. Distributed process management system based on object-centered process modeling. In *Proceedings of 2nd International Conference on Worldwide Computing and Its Applications '98*, pages 108–119, 1998.

[11] K. Ochimizu. Survey of Research Activities on Software Process. *Journal of Information Processing Society of Japan*, 36(5):379–391, 1995.

[12] S. Sutton Jr., D. Heimbigner, and L. Osterweil. APPL/A: A Language for Software Process Programming. *ACM Transactions on Software Engineering and Methodology*, 4(3):221–286, 1995.

[13] P. Tarr and L. Clarke. PLEIADES: An Object Management System for Software Engineering. In *Proceedings of the First ACM SIGSOFT Symposium on the Foundations of Software Engineering*, volume 18, pages 56–70, 1993.

[14] R. Taylor, F. Belz, L. Clarke, L. Osterweil, R. Selby, J. Wileden, A. Wolf, and M. Young. Foundations for the Arcadia Environment Architecture. In *Proceedings of 3rd ACM SIGSOFT/SIGPLAN Symposium on Practical Software Development Environments*, pages 1–13, 1988.