

Effective Reuse Procedure for Open Source Software

Doo Yeon Kim

Ministry of Education
& Human Resource Development
Seoul, Korea

Jong Bae Kim

Department of Computer Science
Soongsil University
Seoul, Korea

Sung Yul Rhew

Department of Computer Science
Soongsil University
Seoul, Korea

Abstract - *In recently, OSS may use to make a new alternative substitute for software development. However, detailed procedures and methods to utilize OSSs are inadequate. When we develop an application based on OSS, we may not apply to the traditional process models. Therefore, newly defined procedures have to be required to reuse the OSS.*

In this paper, we propose the comprehensive procedures based on Meng Huang's process, component based development and related research about OSS. We define 4 steps and 11 activities. To derive an effectiveness and improvement for each activity, we may apply proposed procedures to real projects.

Keywords: Open Source Software (OSS), Reuse, Development Procedures based on OSS

1 Introduction

Recently, open source softwares (OSSs) are widely used in various domains through open, copy, revise, and release. The companies are trying to apply to software development approach by utilizing open source software. The OSSs are new alternatives to solve the limits of the previous software developments such as quality of software, time and cost of development. Accordingly, various aspect analyses of OSSs should be performed. However, the researches about the detailed procedures and methods to utilize OSSs in practical industry are immature [1][2][3][4][5][6].

Most of OSS-Based Applications (OBA) are similar to COTS (Commercial-Off-The-Shelf) based on development of COTS-Based Applications (CBA). That is, suitable selections of OSS or COTS, modification, integration are similar. However, detailed procedures between OBA and CBA development are different. When we develop based on OSS, we use the directly modification but CBA is not use the directly modification since COTS doesn't provide a source code. When we develop an application based on OSS, we may not apply to the traditional process models. Therefore, newly defined procedures should be required to reuse the OSSs.

In this paper, we propose the comprehensive procedures based on Meng Huang's process, component

based development about OSSs. We define 4 steps and 11 activities. To derive an effectiveness and improvement for each activity, we apply proposed procedures to real projects.

2 Related Work

2.1 T.R. Madanmohan & Rahul De's Work

This work [7] proposed the selection model of open source component. This model is composed of three basic steps as collection, incubation and critical revision. This work restricted in selection of open source component. Also, this work is able to apply to overall selection of OSSs. However, detailed procedures and activities to utilize this work are required. And, this work needs to extend overall OSS development not restricted component development.

2.2 Meng Huang' work

This paper [8] offered a development process of building open source based application that is analogous to the process of developing component based application. The proposed process emphasized the early assessment to improve the architecture stability and project manageability by assessing available OSS. However, Meng Huang' work has limitations to utilize overall procedure in application based on modification and integration of OSSs. Also, this paper described the artifacts, but relationships between artifacts are not specified.

3 Reuse Procedures of OSSs

In this section, we propose the reuse procedures of OSSs that based on Ming's research and component based development. Reuse procedures are composed of 4 steps and 11 activities. Also, these procedures define input/output artifacts for each step and activity. As shown in figure 1, 4 steps and 11 activities are defined.

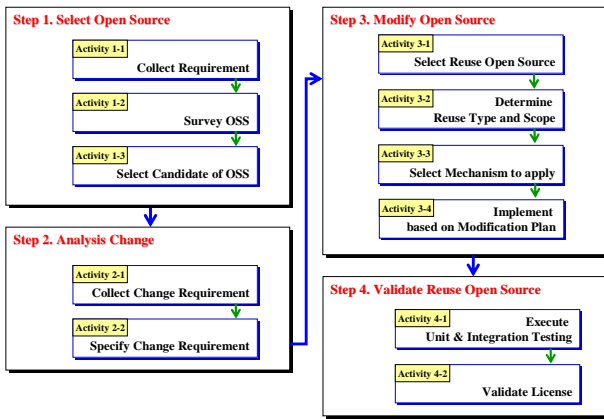


Figure 1. Reuse Procedures of OSSs

3.1 Step 1. Select Open Source

Presently, open sources are provided by many communities and sites. Therefore, step to select suitable candidates from various open sources are preceded. As shown in the figure 2, this step defines three activities for identifying the correct requirements for target software and for surveying OSS, and for selecting candidate OSS.

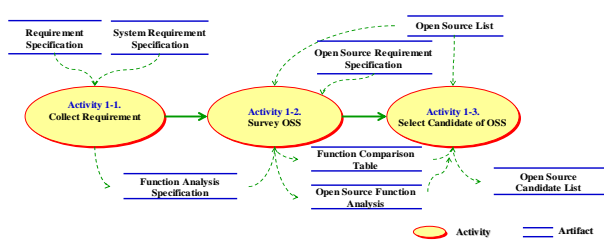


Figure 2. Activities of Step 1

3.1.1 Activity 1-1. Collect Requirement

This activity is to analyze the functional and nonfunctional requirements of target software. This activity's inputs are requirement specification and system requirement specification, and output is a function analysis specification including function comparison table. Function analysis specification is used to survey OSS for reuse as a baseline.

3.1.2 Activity 1-2. Survey OSS

This activity surveys main open source site and community, examine reusable software based on requirement specification. This activity's inputs are open source list and open source requirement specification, and outputs are function comparison table and open source function analysis specification. Open source function analysis specification is not detailed performed for comparing functions between OSS and target software.

3.1.3 Activity 1-3. Select Candidate of OSS

This activity is to select candidate of OSS based on function comparison table. This activity's inputs are open source list, open source function analysis specification and function comparison table, and output is open source candidate list. In this activity, criteria for selecting open source candidate are established. These criteria have different selection standard by discretion of project leader. Generally, since it is difficult to find that open source softwares are satisfied with all required functions, we may determine the criteria as 70% and over by discretion of project leader. Therefore, these criteria are different from characteristics of project and discretion of project leader.

3.2 Step 2. Analysis Change

It is difficult to find fully consistency OSS satisfying requirement specification since characteristics and objectives of target software are various. Therefore, candidate OSS may use not without modification and use through modification of partially functions. To modify and reuse of candidate OSS, identifying and analyzing of change requirements are required. As shown in figure 3, these activities are defined.

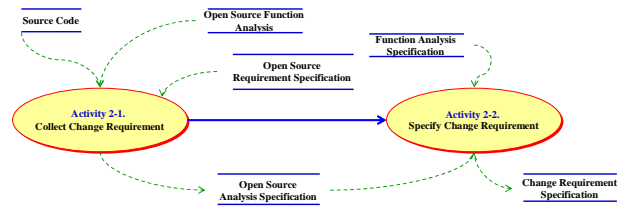


Figure 3. Activities of Step 2

3.2.1 Activity 2-1. Collect Change Requirement

OSSs may provide the partially specification and no specification. Therefore, we may analyze the change requirement using only source codes. This work is a very not easy. In this activity, detailed information of candidate OSSs should be collected. To do this, we can reference evaluations and opinions of the OSS experienced user through communities as forums and news are provided by open source sites or homepages are operated by project teams or companies of relevant open source. This activity's inputs are open source function analysis specification, open source requirement specification and source code, output is open source analysis specification. Open source analysis specification includes the detailed information such as input value, output value, parameter type of functions.

3.2.2 Activity 2-2. Specify Change Requirement

This activity is to specify through analyzing the change requirement. Change requirement is a specification of changeable part for reusing each candidate open source.

This activity's inputs are open source analysis specification and function analysis specification, and output is a change requirement specification.

3.3 Step 3. Modify Open Source

After evaluating the candidate open source list based on various elements, selection of reusable OSS and determination of type, scope and mechanism of selected OSS are performed. Activities for modification of open source are defined in figure 4.

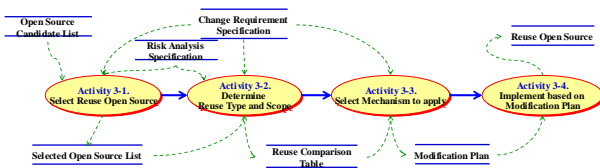


Figure 4. Activities of Step 3

3.3.1 Activity 3-1. Select Reuse Open Source

This activity's inputs are open source candidate list, change requirement specification and risk analysis specification, and output is a selected open source list. To select an open source using reuse through change, risk and cost by changing or modifying are considerate. That is, we may select the open source having a minimum cost and maximum effectiveness after risk analysis when open source are change.

3.3.2 Activity 3-2. Determine Reuse Type and Scope

Since open source are reused by various types, reuse types should be determined. Selected reuse open source partially reused as one function or class. Therefore, scope of reusable open source is determined. This activity's inputs are risk analysis specification and change requirement specification, and output is a reuse comparison table.

3.3.3 Activity 3-3. Select Mechanism to apply

After determined reuse type and scope, mechanism for changing is also determined. This activity's inputs are change requirement specification and reuse comparison table, output is a modification plan. Reuse type of open source is a source code reuse, framework reuse, and component reuse [9]. According to this type, detailed activities are divided. However, we don't describe the detailed activities following reuse type.

Source code reuse. Source code reuse is the simplest case of reuse possible, because it doesn't use any particular mechanism. However, source code reuse use when part of program is a useful and but not individually packaged. In

this case, code reuse needs to suitable change from copied a part of code. However, one advantage acquired from component reuse such as maintenance of reused coded is missed. Therefore, only a little source code has to change, reference code instead of code copy, use inheritance mechanism for modification.

Component reuse. Since COTS component provides only a binary code and specification, change of code are not performed. However, since inner code of open source is known, direct change of source code may perform. Hence, because this modification makes low an effectiveness of component, modification of component is restricted performed. Therefore, customization or connector without directly modification is used.

Customization [11] uses to change the attribute and workflow of open source component. Attribute of open source component is changed by getter or setter function of interface. When workflow is changed by interface, class name of workflow transmit to interface of open source component.

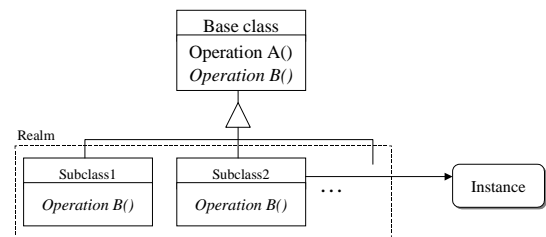


Figure 5. Example of Customization Mechanism

Connector [11][12] uses to solve the mismatch such as association or dependency between existing component and open source component. In the figure 6, the open source component requests a method invocation on the existing component through the connector. The request is delivered to the connector through port, and the connector performs a designated role such as data transformation or interface mediation, and transmits a modified request to the target component through port.

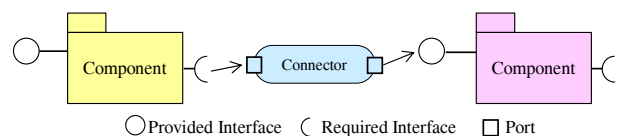


Figure 6. Example of Connector Mechanism

Framework reuse. Framework is to allow its users to reuse an existing structure providing different functionalities or coding advices across different programs. It can be considered a kind of software reuse, similar to components reuse, but bringing different functionalities together in a single framework.

3.3.4 Activity 3-4. Implement based on Modification Plan

This activity's input is a modification plan and output is a reuse open source. In this activity, real modification of open source performs with modification plan.

3.4 Step 4. Validate Reuse Open Source

This activity is to execute testing to validate not only unit testing caused by changing open source but also integration testing to validate whether or not has a problems when integrate between modified reuse open sources and existed sources. This activity defines executing unit and integration testing and validating licenses.

3.4.1 Activity 4-1. Execute Unit & Integration Testing

This activity validates elements such as testing of modified source code, integration testing between existing code and modified code, and integration testing between reuse open sources.

3.4.2 Activity 4-2. Validate License

To utilize the open source, we have to need a compliance of license. After development and test are finished, the developed final artifact as source code is need to validating. In spite of development plan with license issue, it is possible to using in development step without validating license.

4 Case Study

In this section, we applied the proposed procedures to project as supporting tool for small scale software development methodology based on OSS. First, we identified functions from the requirement for surveying reusable open source. The main functionalities of supporting tool domain are following as the artifact management, schedule management, resource management, etc. The total number of functions described in this domain was 30. Also, identified non-functional were 10. Next, we specified the function analysis table to survey suitable OSS for reuse. We surveyed the well-known open source site as <http://sourceforge.net> using keywords as functions through function analysis table. Surveyed list is categories including *Project Management*, *Groupware*, and *Community* [13][14]. Open source sites provided the following information as briefly function descriptions, Project Details, Development Status, License, Registered, Activity Percentile, Operating System, Programming Language, and User Interface. As shown in table 1 is a function comparison table.

Table 1. Function Comparison Table

OSS	Target Software								Conformance
	F01	F02	F03	F29	F30	
GW01	✓						✓	✓	10%
GW02				✓	✓	✓	✓	✓	70%
GW03		✓						✓	50%
...									
PM01	✓	✓	✓		✓	✓	✓	✓	85%
PM02	✓	✓			✓	✓		✓	67%
...									
Comu01			✓		✓	✓		✓	77%
Comu01	✓			✓		✓		✓	52%
...									

Through collecting and specifying the detailed information of candidate open source, we specified the change requirement specification of each candidate open source. The table 2 is an example of PM01's *change requirement specification*.

Table 2. Change Requirement Specification of PM01

Function		Agreement/Disagreement	Change Requirement
Function	F01.Artifact Management	Disagreement of Parameter	Parameter Coherence
	F02.Schedule Management	Disagreement of Operation	Operation Modification
	F03.Resource Management	Disagreement of Data	Data Coherence
	None	None

To examine numerous source codes for eliciting information from OSS only having source code without specification has a limitation. Therefore, it is required to using re-engineering tool, which elicit from source code to specification information. Also, we selected 3 reusable open sources among open source candidate list based on risk and cost caused by change. We determined that PM01 as project management function integrated with GW02 as module management, schedule management, web mail function. To do this, we selected reuse type and scope through comparison of reuse possibility about selected open source. The table 3 is an example for comparison table of PM01.

Table 3. Comparison Table of PM01

Function		Reuse Type	Reuse Possibility
Function	F01.Artifact Management	Source Code	O
	F02.Schedule Management	Source Code	X
	F03.Resource Management	Framework	O
	...	Component	O

Selected open source was not satisfied to requirement of target software. Therefore, we performed design and implementation through analyzing architecture and source code. Also, we defined interfaces of user and schedule

management, and integrated new function and revised source code.

We experienced the continuously change and extend during modification of open source. Therefore, we need to determine change of modified project and complement of preliminary procedures when design and develop without repetition of the existing activities.

5 Conclusion

OSS based software development is a differ from general software reuse in some point selecting open source, collecting change requirement, determining reuse type and scope, and validating license.

Our paper proposed the 4 steps and 11 activities for software development procedures to utilize OSSs. By applying the proposed procedures, we reduced the development time to market.

However, we may study the metric for correctly and objective to select the reuse open source and need the detailed definition of procedure and mechanism according to reuse type. In the future work, we will perform the study on additional researches for more detailed and more practical reuse process development, and applying to reuse process for OSSs.

6 References

- [1] Research, FLOSS(Free/Libre and Open Source Software: Survey and Study), FINAL REPORT, 2002.
- [2] MICHAEL J. KARELS, "Commercializing Open Source Software", ACM Queue, vol.1, no.5, 2003.
- [3] Feller, J. and Fitzgerald, B., "A Framework Analysis of the Open Source Software Development Paradigm", Proceedings of the twenty first international conference on Information systems table of contents, pp.58-69, 2000.
- [4] James W. Paulson, Giancarlo Succi, Armin Eberlein, "An Empirical Study of Open-Source and Closed-Source Software Products", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, Vol.30, No.4, 2004.
- [5] Michel Ruffin and Christof Ebert, "Using Open Source Software in Product Development: A Primer", IEEE Software, vol.21, no.1, pp.82-86, 2004.
- [6] Diomidis Spinellis and Clemens Szyperski, "How is Open Source affecting software development?", IEEE Software, Vol. 21-1, pp. 28–33, 2004.
- [7] T.R. Madanmohan and Rahul De', "Open Source Reuse in Commercial Firms", IEEE Software, Vol. 21, No. 6. pp. 62-69, 2004.
- [8] Meng Huang, Liguang Yang, and Ye Yang, "A Development Process for Building OSS-Based Applications", SPW 2005, LNCS 3840, pp.122–135, 2005.
- [9] ODETTE Project, Software Reuse in Free software: State-of-the-art, 2005.
- [10] Keepence, B., and Mannion, M., "Using patterns to model variability in product families," IEEE Software, Vol. 16, Issue. 4, pp. 102-108, July-Aug., 1999.
- [11] Clements, P., et al., Documenting Software Architectures: Views and Beyonds, Addison-Wesley, pp. 112-113, 2003.
- [12] D'Souza, D. F., and Wills, A. C., Objects, Components, and Frameworks with UML, Addison-Wesley, pp. 477-478, 1999.
- [13] Sourcefoge.net, <http://www.sourceforge.net>
- [14] Freshmeat.net, <http://www.freshmeat.net>