LETTER *Special Section on Log Data Usage Technology and Office Information Systems*

Effects of Software Modifications and Development After an Organizational Change on Software Metrics Value

Ryo ISHIZUKA[†a)], **Naohiko TSUDA**[†], ***Nonmembers***, **Hironori WASHIZAKI**[†], **Yoshiaki FUKAZAWA**[†], ***Members***, **Shunsuke SUGIMURA**[††], ***and*** **Yuichiro YASUDA**[††], ***Nonmembers***

SUMMARY Deterioration of software quality developed by multiple organizations has become a serious problem. To predict software degradation after an organizational change, this paper investigates the influence of quality deterioration on software metrics by analyzing three software projects. To detect factors indicating a low evolvability, we focus on the relationships between the change in software metric values and refactoring tendencies. Refactoring after an organization change impacts the quality.

key words: *software quality, software metrics, organizational change, outsourcing*

1. Introduction

Software development often involves multiple organizations. A common cost-savings measure is to outsource part of software development, even though software developed by multiple organizations tends to have a lower quality. This is because software development with multiple organizations often leads to differences in recognizing software specifications, poor change controls, and inconsistencies in the software architecture [1], [2].

The influences of organizational changes have also been shown. Sato et al. [3] found that source files developed by multiple organizations tend to be more faulty than other files from two open source projects.

Our research aims to detect factors that negatively influence software developments by multiple organizations. To capture the changes in software from various angles, we focus on software metrics and editing tendencies of the source code. Previously we analyzed the characteristics of source codes developed by multiple organizations [4]. This paper narrows the relationships between the change in the software metric values and the editing tendencies after an organizational change.

2. Case Study Design

In this paper, we analyze the influence of refactoring and adding features on software quality after an organizational change. Table 1 summarizes the software subjects. Tool A and Tool B are internal and outsourced tools with a change in the subcontractor. OpenOffice.org (OOo) is an office suite developed by Sun MicroSystems, which branched into two successor software programs: LibreOffice (LibO) and Apache OpenOffice (AOO). LibO was developed by the Document Foundation, which was founded by OOo developers. AOO was developed by the Apache Software Foundation.

Table 1 Details of the experimental subjects

Software		Development Organization		Releases	
Original	Successor	Original	Successor	Original	Successor
Tool A		Company A	Company A	2	2
Tool B		Company B		2	2
OOo	LibO	Sun Microsystems	TDF	2	13
	AOO		ASF		9

To capture the changes in quality, we measured five software metrics and the number of software defects. The software metrics were selected to realize a multifaceted evaluation. Direct Fan Out (DFO) and Visibility Fan Out (VFO) were used to assess the file dependency [5]. VFO captures the changes in the software architecture by counting both direct and indirect dependencies between source files. Moreover, we selected Lines of Code (LOC) to find complex files, Ratio of Code Clones to maintain code clones, and Ratio of Comments to evaluate readability. These metrics are generally used as simple software metrics. We used a static analysis tool (Understand) [6] and two code clone detection tools [7], [8] to measure these metrics. Tables 2 and 3 show the transition of the software metric values of subjects.

To detect deteriorated software quality, we identified modified files and counted the number of defects modified in each file after an organizational change. To count the number of defects of Tool A and Tool B, we manually identified modified files with defect information in the development history. We found about 3.54 defects per file in 28 source files of Tool A and about 3.24 defects per file in 21 files of Tool B. To count the number of defects of LibO and AOO, we used the szz algorithm [9] from the git commit log. We found about 3.48 defects per file in 22,728 files in LibO from 129,916 commits and about 0.15 defects per file in 23,386 files in AOO from 5,824 commits.

To capture the editing tendency of the source code, we tracked how the source code was modified by refactoring

Manuscript received November 12, 2018.
Manuscript revised April 7, 2019.
Manuscript publicized June 13, 2019.

[†]The authors are with Department of Computer Science and Engineering, Waseda University, Tokyo, 169–8555 Japan.
[††]The authors are with ICT Development Center, Komatsu Ltd., Hiratsuka-shi, 254–0014 Japan.
a) E-mail: ryo_issy@fuji.waseda.jp

DOI: 10.1587/transinf.2018OFL0004

Table 2 Transition of the software metric values of the final version of each series of OOo, LibO, and AOO

version	OOo 3.3.0	LibO 3.6.7.2	LibO 4.4.7.2	LibO 5.4.4.1	AOO 3.4.1	AOO 4.0.1	AOO 4.1.4
LOC	4.786×10^6	4.721×10^6	4.768×10^6	4.677×10^6	5.556×10^6	4.837×10^6	4.849×10^6
DFO	4.044×10^5	3.933×10^5	3.943×10^5	3.926×10^5	4.798×10^5	4.151×10^5	4.139×10^5
VFO	1.556×10^8	1.451×10^8	1.719×10^8	1.795×10^8	2.809×10^8	1.760×10^8	1.709×10^8
RatioOfCodeClone	0.118	0.108	0.108	0.096	0.176	0.099	0.098
RatioOfComment	0.206	0.200	0.157	0.157	0.221	0.190	0.190

Table 3 Transition of the software metrics values of Tool A and Tool B

Release	Tool A 1st	Tool A 2nd	Tool A 3rd	Tool A 4th	Tool B 1st	Tool B 2nd	Tool B 3rd	Tool B 4th
LOC	2300	2589	3610	5626	1775	2088	4365	4895
DFO	84	74	75	84	36	38	64	62
VFO	129	144	183	184	57	59	137	123
RatioOfCodeClone	0.198	0.148	0.081	0.070	0.013	0.040	0.063	0.053
RatioOfComment	0.254	0.243	0.238	0.176	0.276	0.269	0.256	0.251

Table 4 Changes of module "sw" of LibO

Release	Changes of the module at release
~3.3.99.5	1. Insert new coding rules 2. Unify code format
~4.0.7.2	1. Delete unnecessary comments 2. Reduce Macros whose name are too long
~5.0.6.3	1. Format comments to simplify 2. Translate comments in English 3. Rename class names to avoid acronyms
~5.4.4.1	1. Introduce type inference to rewrite a repetitive statement 2. Rewrite loop to range-based

and adding features. We manually identified these at each release and compared whether each fix or refactoring affected the metrics value. Table 4 show the editing tendency of one module in LibO, which we conducted code review.

3. Case Study Results

In LibO, the value of DFO, the ratio of code clones, and comments decreased. However, the LOC remained fairly constant although new software features were added. On the other hand, the value of VFO increased from version 4 but DFO did not. Several refactorings such as the arrangement of macros and the loop statements affected LOC. As for the ratio of comments, the fact that the format of the comments changed in version 4 series contributed a lot. As for DFO and VFO, we considered that the software architecture was redesigned prior to version 4, and the software dependency became more complicated by adding new features.

The source code did not differ from AOO 4.1.0 because the number of the commits for AOO decreased. In 2012, there were 1927 commits for AOO per year, but only 204 commits in 2017. On the other hand, LibO had 19283 commits in 2012 and 15922 commits in 2017. We attributed this difference to the two reasons. First, Sun MicroSystems postponed the refactoring of OpenOffice for many years. Second, most of the OOo developers were involved with LibO, but few were involved with AOO. Then, the difference between LibO and AOO is the transition of the VFO values. Hence, the complexity of the software architecture negatively influences the understanding of the software for the AOO developers.

Tool A and Tool B did not confirm that software architectural refactorings such as method extraction were performed. However, individual files such as indentation and rewiring logic of the methods were refactored. Consequently, the number of features in one file increased due to the addition of the new features in subsequent releases. Hence, LOC and the depth of the nest increased but the readability of such methods decreased.

4. Conclusion

This paper analyzed the origin of quality degradation using software metric values and the editing tendencies after an organizational change. First, the complexity of the software negatively influenced the understanding of the software. Second, the existence of refactoring after an organizational change positively influenced the metric value. Thus, refactoring should be performed after an organizational change to maintain the software architecture. However, we cannot judge the universality of this evaluation as only three software were assessed. In the future, we will collect new software subjects.

References

[1] M.E. Conway, "How do committees invent?," Datamation, vol.14,

no.4, pp.28–31, 1968.
[2] R.T. Nakatsu and C.L. Iacovou, "A comparative study of important risk factors involved in offshore and domestic outsourcing of software development projects: A two-panel Delphi study," Information and Management, vol.46, no.12, pp.57–68, 2009.
[3] S. Sato, H. Washizaki, Y. Fukazawa, S. Inoue, H. Ono, Y. Hanai, and M. Yamamoto, "Effects of Organizational Changes on Product Metrics and Defects," Proc. 20th Asia Pacific Software Engineering Conference (APSEC 2013), pp.132–139, 2013.
[4] R. Ishizuka, N. Tsuda, H. Washizaki, Y. Fukazawa, S. Sugimura, and Y. Yasuda, "Characteristics of unmaintainable source code in software development by multiple organizations," 3rd IEEE/ACIS International Conference on Big Data, Cloud Computing, and Data Science Engineering (BCD 2018), pp.49–54, 2018.
[5] D.J. Sturtevant and D. Joseph, "System design and the cost of architectural complexity," in Thesis (Ph.D.) Massachusetts Institute of Technology, Engineering Systems Division, 2013.
[6] Scientific Toolworks, Inc., "Understand," http://www.scitools.com/, Last Accessed Oct. 2018.
[7] Nicad Clone Detector, https://www.txl.ca/txl-nicaddownload.html, Last Accessed March 2019.
[8] PMD's CPD, http://pmd.sourceforge.net/pmd-4.3.0/cpd.html, Last Accessed Oct. 2018.
[9] J. Sliwerski, T. Zimmermann, and A. Zeller, "When do changes induce fixes?," Proc. 2nd International Workshop on Mining Software Repositories (MSR 2005), pp.1–5, 2005.