Effects of Organizational Changes on Product Metrics and Defects

Seiji Sato, Hironori Washizaki,
and Yoshiaki Fukazawa
Department of Computer Science
and Engineering
Waseda University
Tokyo, Japan
Email: r0d8h8i0h@asagi.waseda.jp,
{washizaki, fukazawa}@waseda.jp

Sakae Inoue, Hiroyuki Ono,
Yoshiiku Hanai, and Mikihiko Yamamoto
Fujitsu Limited
Kanagawa, Japan
Email: {inoue.sakae, ono.hiro, hanai.yoshiiku,
yamamoto.mikihi}@jp.fujitsu.com

***Abstract*—The development organization often changes during software development. Derivative developments, forks, and change of developers due to acquisition or open-sourcing are some conceivable situations. However, the impact of this change on software quality has yet to be elucidated. Herein we introduce the concept of *origins* to study the effects of organizational changes on software quality. A file's origin is defined as its creation and modification history. Using the concept of origins, we analyze two open source projects, OpenOffice and VirtualBox, which were each developed by a total of three organizations. We conduct statistical analysis to investigate the relationship between the origins, product metrics, the number of modifications, and defects. Results show that files that are modified by multiple organizations or developed by later organizations tend to be faultier due to the increase in complexity and modification frequency.**

***Keywords*—Organizational change, product metrics, defects**

I. Introduction

A software program is not always developed by a single organization. Several organizations may co-develop software, or a different organization may take over the development before the software is completed. Derivative developments, forks, and a change of developers due to an acquisition or open-sourcing are some conceivable situations.

Organizational changes may reduce software quality. This is because an organizational change significantly changes the structure of the organization, and the change in the structure leads to reduction of the software quality. Conway's Law [1], [2] states that software design reflects the structure of its development organization. Because different organizations have different structures, an organizational change may lead to discrepancies. Furthermore, Mockus et al. [3] showed that developers leaving a project create gaps in tacit knowledge, reducing the software quality. Therefore, an organizational change, which often involves replacing many of the developers, can cause a significant loss of software quality.

If a software file is developed by several organizations, the organizations can be divided into those that modified the file and those that did not. This work uses this characteristic to propose a method to analyze the effects of changing the development organization. We introduce a metric called *origins*, which is defined as the creation and modification history of a file.

In our study, we hypothesize that organizational changes affect the reliability of the software, hence the number of defects. Therefore, we examine the relationship between origins and defects. We additionally hypothesize that the relationship between origins and defects is caused by the change of the code complexity, which is measured by product metrics related to maintainability, and the modification frequency. Previous works have shown that product metrics [4], [5], [6], [7], [8] and code modifications [9], [10], [11] are good indicators of software faults. Hence, we investigate the relationships between origins, product metrics, the number of file modifications, and defects.

Some origins may lower software quality. Previous works have found that recently changed modules and frequently changed modules tend to have a lower quality [9], [10], [11]. Additionally, Lim [12] has shown that reused code (i.e., code created previously and not modified) has less defects than new code. From these works, we hypothesize that modules modified by multiple organizations or developed by later organizations tend to be of lower quality.

Thus, this work investigates the following research questions:

- RQ1 Are files modified by multiple organizations or developed by later organizations more faulty?
- RQ2 How do product metrics relate to origins and defects?
- RQ3 How does the number of modifications relate to origins, defects, and product metrics?

Figure 1 shows the relationships between origins, product metrics, the number of modifications, and defects, that the research questions investigate.

To address the above-mentioned research questions, we propose origin and the method to analyze software by using origin. In our study, we analyze two open source projects, OpenOffice[1] and VirtualBox[2], and find the relationships be-

[1] http://www.openoffice.org/
[2] https://www.virtualbox.org/

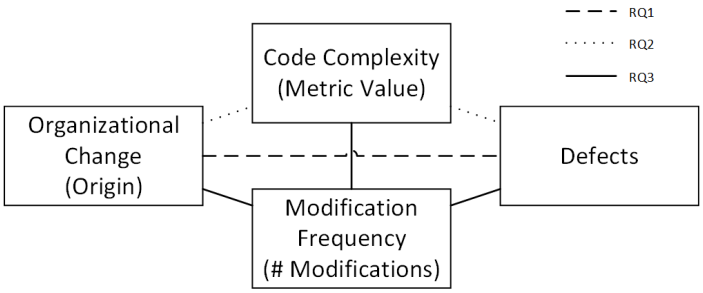


Fig. 1: Relationships between the origins, product metrics, the number of modifications, and defects investigated by the research questions

tween origins, product metrics, the number of modifications, and defects.

1) A method to analyze the effects of organizational changes on software whose development organization has changed several times.
2) Analysis of practical two open source projects to find the concrete relationships between origins, product metrics, the number of modifications, and defects. More precisely,
   a) An investigation on the relationship between origins and defects, finding that files modified by multiple organizations or developed by later organizations are faultier.
   b) An investigation on the relationships between the origins, product metrics, and defects, finding that files modified by multiple organizations or developed by later organizations are more complex, and consequently have more defects.
   c) An investigation on the relationships between the origins, number of modifications, and defects, finding that a difference in the original file causes a difference in the number of modifications, which consequently causes defects.

II. Background

A. Effects of Organizational Change on Quality

As discussed in Section I, organizational changes can impact software quality. Although several studies have employed organizational metrics to analyze the organizational aspects of software development, their focus differs from ours.

Nagappan et al. [13] used eight metrics mainly based on organizational structure to demonstrate that organizational metrics are superior to code churn, code complexity, code coverage, code dependencies, and pre-release defect measures. To investigate the influence of development by multiple organizations, they used a metric that measures the number of different organizations involved in the development of a module. However, this metric only observed the number of organizations and did not consider the detailed changes of the organizations.

To confirm the effectiveness of predicting fault proneness, Mockus [3] used five metrics that focused on changes in the developers within an organization. However, their method did not evaluate changing organizations.

Graves et al. [10] constructed and evaluated fault prediction models for three module categories: modules changed by organization A, those changed by organization B, and those changed by both. Although classifying the source code based on which organizations modified it is similar to ours, their method is not for cases where different organizations successively developed the software. Furthermore, our method is more generalized, and can deal with source code developed by three or more organizations.

B. Motivating Example

Several software projects have had multiple organizational changes while being developed. One example is OpenOffice, which was developed by three organizations: Sun Microsystems, Oracle Corporation, and Apache Software Foundation. (Strictly speaking, OpenOffice originated from StarOffice, which was developed by StarDivision.) OpenOffice was initially developed by Sun, and was transferred to Oracle after Oracle acquired Sun. Later, Oracle contributed OpenOffice to the Apache Software Foundation. These organizational changes may have affected the software quality. Although we hypothesize that modules modified by multiple organizations or developed by later organizations are of lower quality (Section I), a method to analyze this scenario has not been reported, and the influence of multiple organizations on software quality has yet to be elucidated.

III. Analysis of Software Quality Using Origins

A. Origin of a File

To analyze the impact of development organizational changes, determining which modules are affected by this type of change is important. Three types of files exist once a different organization takes over software development: files that are created by the organization, files that are modified by the organization, and files that are not modified by the organization. An organizational change will impact the first two types of files (created or modified), but not the latter (unmodified). Created files are affected because newly created modules are based on existing software, and many of the existing modules are created by previous organizations.

Origins can be used to classify the software files by the organization that created or modified them. If n organizations successively developed the software, the origin of a file in the software is denoted by $O_{t_1 t_2 \ldots t_m}$, where $t_1 t_2 \ldots t_m$ represents the sequence of the organizations that modified the file, and $1 \leq m \leq n$. Note that the organizations are ordered sequentially, and that the first term in the sequence of organizations corresponds to the organization that created the file. Deleted files are not considered here because they are not included in the final product.

Figure 2 shows an example of the origins involving three organizations. For simplicity, we denote the origin as O_{123} rather than $O_{Organization_1 Organization_2 Organization_3}$. After the first organization develops the software, there is only one origin, O_1, which is created by that organization. After the development by the second organization, there are three origins: O_1 (created by the first organization and not modified by the second organization), O_{12} (created by the first organization and modified by the second organization), and O_2 (created by the

second organization). Furthermore, after the third organization develops the software, there are total of seven origins.

Section III-B1 describes how our method obtains the origin.

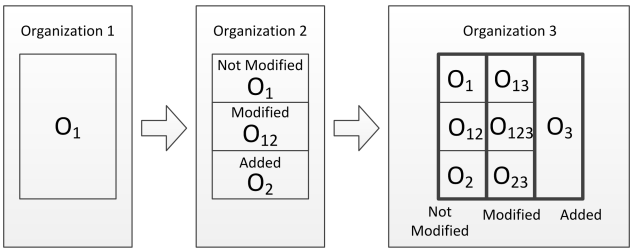


Fig. 2: Origins when three organizations are involved in software development

B. Analysis of Origins, Product Metrics, and Defects

As previously discussed, because the modification history varies by organization, origins differ. Thus, the origin can possess different characteristics in terms of quality. We have analyzed the effects of the origin on the product metrics and file defects in software projects.

Figure 3 shows the analysis process. (i) First, the origins, product metric values, defect existence, and the number of file modifications in the target software are determined. To analyze the defects, we use the defects reported after a revision where the product metrics and origins are obtained so that the effects of the origins can be investigated. (ii) Second, the data is combined. (iii) Third, the metric values, defect existence, and the number of modifications of files are classified by origin. (iv) Fourth, statistical analysis of the relationships between the origins, product metrics, defects, and number of modifications is conducted.

1) Determination of the Origins: Although any number of organizations can be involved in software development, for simplicity, here we assume there are only three organizations. We call these organizations org_1, org_2, and org_3, which correspond to the order in which they developed the software.

Figure 2 shows that seven origins are possible with three organizations. The origins of the software files are obtained by the following procedure. Table I shows the summary of the following descriptions, and Figure 4 shows an example.

1) Prepare three snapshots of the source code directory, which correspond to the final products of org_1, org_2, and org_3. Let them be dir_1, dir_2, and dir_3, respectively.
2) Run diff[3] between each two pair combination of the three directories (three total), as shown in the first column of Table I. When running diff between two directories, e.g., dir_1 and dir_2, one of four messages is outputted for each file: "Identical" if if the file is the same in both directories; "Differ" if the file is in both directories, but is modified; "Only in dir_1" if the file is only in dir_1; and "Only in dir_2" if the file is only in dir_2. The possible results are shown in the second column of the table. These results can be classified into one of nine sets as shown in the third column of the table: e.g., if the diff result for a file between dir_1 and dir_2 is "Only in dir2," that file can be classified into either O_2 or O_{23}, so it is classified into $O_2 \cup O_{23}$.
3) Take the intersection of the nine sets obtained in the previous step to find the origins. For example, O_{23} can be obtained by taking the intersection of $O_2 \cup O_{23}$, $O_{23} \cup O_{123}$, and $O_2 \cup O_3 \cup O_{23}$.

[3]http://pubs.opengroup.org/onlinepubs/9699919799/utilities/diff.html

TABLE I: Pair of sets, diff results, and corresponding sets

Pair of directories which the file is in	Diff Result [a]	Set
dir_1, dir_2	Identical	$O_1 \cup O_{13}$
	Differ	$O_{12} \cup O_{123}$
	Only in dir_2	$O_2 \cup O_{23}$
dir_1, dir_3	Identical	O_1
	Differ	$O_{12} \cup O_{13} \cup O_{123}$
	Only in dir_3	$O_2 \cup O_3 \cup O_{23}$
dir_2, dir_3	Identical	$O_1 \cup O_2 \cup O_{12}$
	Differ	$O_{13} \cup O_{23} \cup O_{123}$
	Only in dir_3	O_3

[a]Note that the diff result column does not include all possible results because file deletion is not considered.

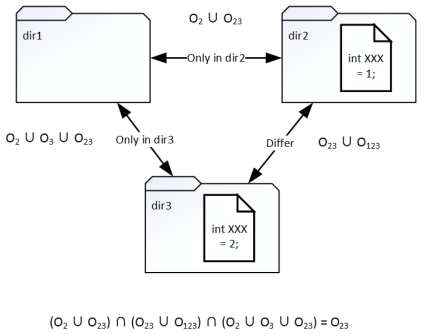


Fig. 4: An example of the determination of an origin

2) Product Metrics: As mentioned in Section I, we need product metrics designed to measure the maintainability of a piece of software to evaluate the complexity of the code. Additionally, as we analyze the relationship with defects, we also use product metrics related to reliability. Here we adopt the GQM approach [14]. GQM approach is a systematic approach to define metrics by associating them with specific goals. We set the goals of our model as evaluating reliability and maintainability. Table II shows the GQM model. Every metrics is considered better if the value is lower. The metrics are from a static analysis tool called Adqua [15]. Because some of the metrics are originally defined in function or class units, we redefine them in file units in order to analyze them with origins, defects, and modifications, which are defined in file units. For example, "number of public methods" (M1) is originally the number of public methods in a class, but is redefined as the total number of methods in the classes in a file.

To measure these product metrics automatically, we created our own tool using a metrics measurement application called Understand [16].

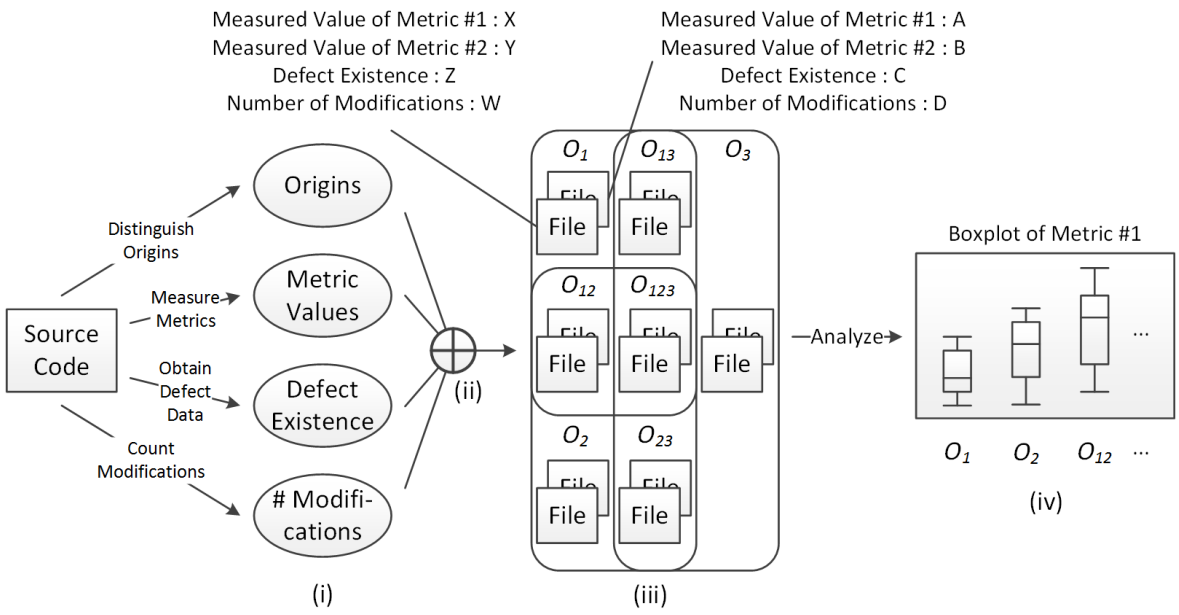


Fig. 3: Analysis Process

TABLE II: GQM model used in this study. All metrics are defined in file units.

Goal	Question	Metric	ID
Evaluate Reliability (Maturity)	Is there no unnecessary accessibility to internal elements?	Number of public methods	M1
		Number of public attributes	M2
	Is the memory space initialized appropriately?	Number of static objects which are not initialized explicitly	M3
Evaluate Maintainability (Analyzability)	Is the code size appropriate?	Physical lines of code	M4
	Is the hierarchical structure appropriate?	Depth of inheritance tree	M5
	Is the abstraction appropriate?	Lack of cohesion in methods	M6
	Are elements concealed appropriately?	Number of global variables	M7
		Number of public static attributes	M8
Evaluate Maintainability (Changeability)	Are there no complex sentences?	Number of lines with multiple statements	M9
Evaluate Maintainability (Stability)	Are effects of external changes limited?	Rate of methods which call methods in other classes	M10
		Number of methods in other classes which this class calls	M11
		Number of functions using global variables defined in other files	M12
		Number of methods using public static attributes defined in other files	M13
		Number of functions and methods defined in other files which this file calls	M14
		Number of global variables defined in other files which this file uses	M15
	Are effects of changes on the outside limited?	Number of global variables used in other files	M16
		Number of public static attributes used in other files	M17
		Number of functions and methods defined in other files which call functions/methods defined in this file	M18

IV. Empirical Evaluation

We analyzed practical software projects using our approach. Here we describe the details of the target project, defect and modification data, and comment deletion.

A. Target Projects

We chose OpenOffice and VirtualBox as target projects. Both are open source projects developed by three organizations. In order, they are Sun Microsystems, Oracle Corporation, and Apache Software Foundation for OpenOffice, and Innotek, Sun Microsystems, and Oracle Corporation for VirtualBox. OpenOffice is written in C++ and Java and has about 27,000 files, while Virtualbox is written in C and C++ and has about 10,000 total files.

After investigating the repositories of these projects, we determined which revisions to use as the snapshots for dir_1, dir_2, and dir_3. We used some of the newest revisions to obtain defect information (Table III). Note that each revision of dir_1 and dir_2 is the last revision of each organization.

B. Defect Data and the Number of Modification

To analyze the influence of the origins on defects, we collected defect information from the Subversion repositories of the target projects. In the OpenOffice repository, the commit messages of most of the defect-correcting revisions have the defect numbers registered in Bugzilla. Therefore, we considered the files modified in those revisions to be faulty. We also included revisions that have “fix” or “correct” in their commit messages so that we do not skip those that do not have defect numbers in their commit messages. In contrast, the commit messages in the VirtualBox repository do not contain

TABLE III: Revision numbers used for each snapshot

Project	dir_1	dir_2	dir_3	Defect Information
OpenOffice	264235 [a]	275822 [a]	1413471 [b]	1414017 - 1488548 (830 Revisions) [b]
VirtualBox	8036	28800	41510	41511 - 46354 (4844 Revisions)

[a]OpenOffice.org repository
[b]Apache OpenOffice repository

defect numbers. Therefore, we only used "fix" and "correct" as indicators. The above-mentioned methods are used in previous work [17], [18], [19].

For the number of modifications, we simply counted the occurrence of files in all revisions.

C. Deletion of Comments

Some files have header comments that contain owner information. However, the header comments change when the owner organization changes, causing a change to the origin despite the fact that no changes have been made to the actual source code. To avoid this, we ignored comments and blank lines when obtaining the origins.

V. Results

Here we show the statistical analysis results of the case study, and discuss our findings with respect to the three research questions.

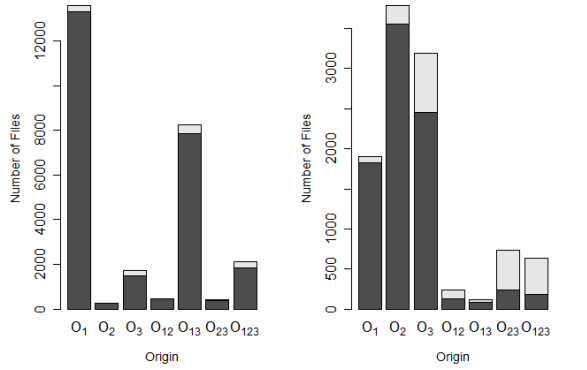
A. RQ1: Are files modified by multiple organizations or developed by later organizations more faulty?

As stated in Section I, files modified by multiple organizations or developed by later organizations may have a lower quality. To investigate this, we determined the relationship between origins and defects.

Table IV summarizes the number of files in each project, which is classified by the origin and the existence of defects. Figure 5 plots the results as bar charts. O_3 and O_{123} in OpenOffice, and O_{23} and O_{123} in VirtualBox have relatively high rate of faulty files. In both projects, all of these files involve many organizations and/or org_3. These results indicate that files modified by multiple organizations or developed by later organizations can be considered to be faultier.

TABLE IV: Number of files in each project

Project	Origin	Total	No Defect	With Defects	Defect Ratio (With Defects / Total)
OpenOffice	O_1	13,604	13,316	288	2.12%
	O_2	250	248	2	0.80%
	O_3	1,728	1,491	237	13.7%
	O_{12}	461	453	8	1.74%
	O_{13}	8,234	7,865	369	4.48%
	O_{23}	409	392	17	4.16%
	O_{123}	2,104	1,836	268	12.7%
VirtualBox	O_1	1,905	1,830	75	3.94%
	O_2	3,793	3,558	235	6.20%
	O_3	3,191	2,457	734	23.0%
	O_{12}	237	128	109	46.0%
	O_{13}	155	82	33	28.7%
	O_{23}	735	238	497	67.6%
	O_{123}	639	180	459	71.8%



(a) Number of files in OpenOffice (b) Number of files in VirtualBox

Fig. 5: Number of files classified by the origins. Black and white bars represent files without and with defects, respectively.

To further clarify this, we classified the origins by the number of organizations and by the last organization to modify the file. Table V summarizes the results. From the results above, we conclude that files modified by multiple organizations or developed by later organizations tend to be faultier.

B. RQ2: How do product metrics relate to origins and defects?

In Section V-A, we demonstrated that origins have a relationship with defects. As mentioned in Section I, when we surmise the cause of the relationship, product metrics is a reasonable candidate. Therefore, we conducted an analysis to investigate the relationships of product metrics to origins and defects.

We first analyzed the relationship between product metrics and origins. We employed box plots of the product metric for each origin, and conducted a Wilcoxon rank sum test for each pair of the origins to test the statistical significance of the differences between the origins. We used the Wilcoxon rank sum test because the Kolmogorov-Smirnov test for each origin of each product metric indicated that none of the samples could be assumed to have a normal distribution.

Among all the product metrics, the most significant one is M14 (Number of functions and methods defined in other files which this file calls). Figure 6 shows the box plots of the metric values classified by the origins. We plotted the box plots without outliers and with outliers to make it easier to see the differences between the origins. Table VI shows the relationship between the samples of the metric values in each pair of origins. For almost all pairs of origins that show

TABLE V: Number of files in each project classified by the origin category

Project	Number of Organizations	Last Organization	Origin	Total	No Defect	With Defects	Defect Ratio (With Defects / Total)
OpenOffice	1	-	O_1, O_2, O_3	15,582	15,055	527	3.38%
	2	-	O_{12}, O_{13}, O_{23}	9,104	8,710	394	4.33%
	3	-	O_{123}	2,104	1,836	268	12.7%
	-	1	O_1	13,604	13,316	288	2.12%
	-	2	O_2, O_{12}	711	701	10	1.41%
	-	3	$O_3, O_{13}, O_{23}, O_{123}$	12,475	11,584	891	7.14%
VirtualBox	1	-	O_1, O_2, O_3	8,889	7,845	1,044	11.7%
	2	-	O_{12}, O_{13}, O_{23}	1,087	448	639	58.8%
	3	-	O_{123}	639	180	459	71.8%
	-	1	O_1	1,905	1,830	75	3.94%
	-	2	O_2, O_{12}	4,030	3,686	344	8.54%
	-	3	$O_3, O_{13}, O_{23}, O_{123}$	4,680	2,957	1,723	36.8%

statistical significance, the p-value is $p < .001$. The only exception is between O_{13} and O_{123} of M14 in VirtualBox ($p < .01$). Several trends are observed for each project.

OpenOffice: O_1, O_2, and O_3 have small metric values. O_{12}, O_{13}, and O_{23} have medium metric values. O_{123} has large metric values.

VirtualBox: O_1, O_2, O_3, and O_{12} have small metric values. O_{13} has medium metric values. O_{23} and O_{123} have large metric values.

In both projects, the origins with many organizations or later organizations tend to be more complex, while in VirtualBox O_{12} does not show the tendency and O_{23} is more complex than O_{123}.

TABLE VI: Relationship between the values of M14 for each pair of origins. $<$ / $>$: Differences between the samples in the origin on the left hand side and those on right hand side side are statistically significant. The former is smaller/larger than the latter. - : Differences between the samples in each origin are not statistically significant.

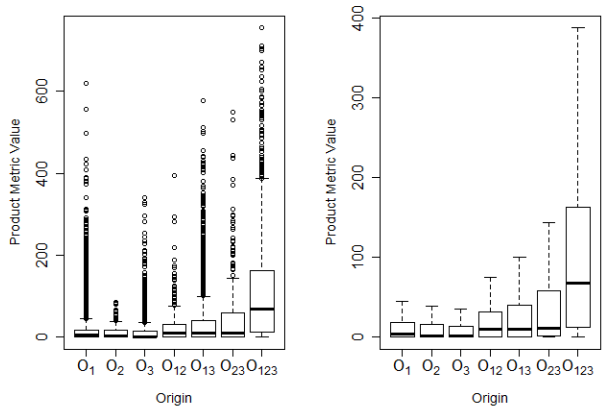
(a) Relations in OpenOffice

Origin on the left hand side	Origin on the right hand side						
	O_1	O_2	O_3	O_{12}	O_{13}	O_{23}	O_{123}
O_1	-	-	$>$	$<$	$<$	$<$	$<$
O_2	-	-	-	$<$	$<$	$<$	$<$
O_3	$<$	-	-	$<$	$<$	$<$	$<$
O_{12}	$>$	$>$	$>$	-	-	-	$<$
O_{13}	$>$	$>$	$>$	-	-	-	$<$
O_{23}	$>$	$>$	$>$	-	-	-	$<$
O_{123}	$>$	$>$	$>$	$>$	$>$	$>$	-

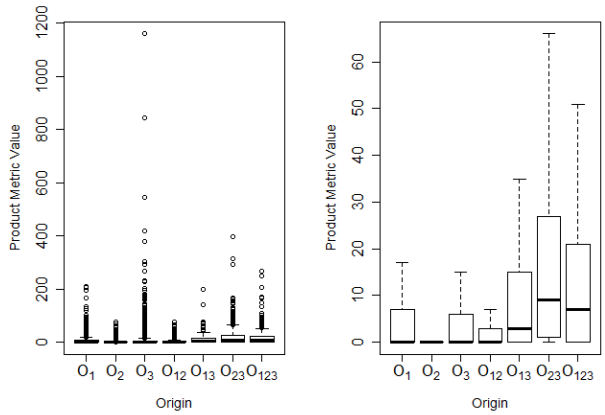
(b) Relations in VirtualBox

Origin on the left hand side	Origin on the right hand side						
	O_1	O_2	O_3	O_{12}	O_{13}	O_{23}	O_{123}
O_1	-	$>$	$>$	-	$<$	$<$	$<$
O_2	$<$	-	$<$	$<$	$<$	$<$	$<$
O_3	$<$	$>$	-	-	$<$	$<$	$<$
O_{12}	-	$>$	-	-	$<$	$<$	$<$
O_{13}	$>$	$>$	$>$	$>$	-	$<$	$<$
O_{23}	$>$	$>$	$>$	$>$	$>$	-	$>$
O_{123}	$>$	$>$	$>$	$>$	$>$	$<$	-

Similar to Section V-B, we classified the origins by the number of organizations and by the last organization, and conducted the same analysis. Figure 7 shows the boxplot without outliers, and shows that files modified by multiple organizations or developed by later organizations tend to have high metric values.



(a) Metric values of OpenOffice with outliers (b) Metric values of OpenOffice without outliers

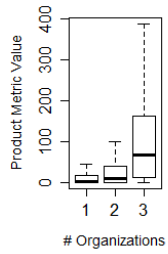


(c) Metric values of VirtualBox with outliers (d) Metric values of VirtualBox without outliers

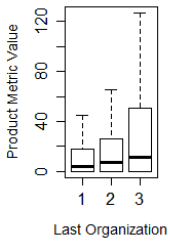
Fig. 6: Box plots of M14 classified by origins

In almost all other metrics, both projects exhibit similar trends as above. Therefore, files modified by multiple organizations or developed by later organizations tend to be more complex.

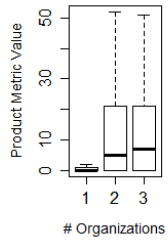
Additionally, we conducted analysis to examine the relationship between product metrics and defects. We compared



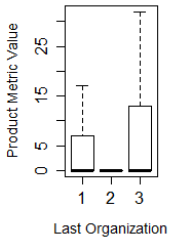
(a) Metric values of OpenOffice classified by the number of organizations



(b) Metric values of OpenOffice classified by the last organization



(c) Metric values of VirtualBox classified by the number of organizations



(d) Metric values of VirtualBox classified by the last organization

Fig. 7: Box plots of M14 without outliers classified by the origin categories

the product metrics values between the files with defects to those without defects. Most of the metrics have higher values in the files with defects than those without defects in both projects.

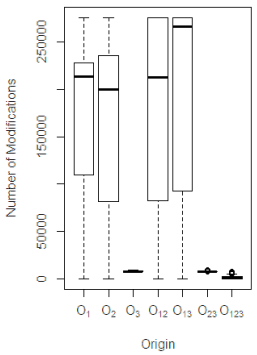
From the above results, we conclude that the product metrics are affected by origin, and consequently affect defects. More specifically, files modified by multiple organizations or developed by later organizations tend to be more complex, and have more defects.

C. RQ3: How does the number of modifications relate to origins, defects, and product metrics?

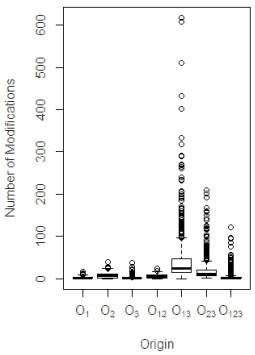
In the previous sections, we examined the relationships between origins, product metrics, and defects. Here we examine the number of modifications and its relationship to the others.

We first analyzed the number of modifications of the files for each origin. Figure 8 shows the results as box plots. In VirtualBox, files modified by multiple organizations or developed by later organizations tend to be modified many times, and in OpenOffice, O_1, O_{12}, O_{13}, and O_{123} have high modification counts. This indicates that the relationships between origins and defects, and origins and product metrics may be intermediated by the number of modifications.

Therefore, to determine whether the number of modifications is correlated with the existence of defects, we calculated the rank correlation coefficient between these two. $\rho = -.013$ for OpenOffice and $\rho = .570$ for VirtualBox,



(a) Number of modifications of files in OpenOffice



(b) Number of modifications of files in VirtualBox

Fig. 8: Number of modifications of files classified by the origins

demonstrating that the impact of the number of modifications is not negligible in VirtualBox. We also calculated the rank correlation coefficient using the number of defects instead of the existence of defects, but the difference is negligible. Accordingly, for VirtualBox, we calculated the partial rank correlation coefficient between each origin and defects, and between the number of modifications and defects, using each origin and the number of modifications as the control variables, respectively. $|\rho| < .2$ for each origin, and $.5 < \rho < .6$ for the number of modifications, indicating that the number of modifications is the main factor to affect the defects in VirtualBox.

Additionally, we calculated the Spearman's rank correlation coefficient between the number of modifications and each product metric value to determine whether they are related. However, for all of the metrics, the absolute values of the rank correlation coefficients are low (most are less than .1, and the others are .1 to .3) in both projects, suggesting that the origins are the main factor to affect the complexity of code.

From the above results, we conclude that the number of modifications is related to origins and defects depending on the project, but relates slightly with the product metrics.

Figure 9 shows the relationships between all four factors. The files modified by multiple organizations or developed by later organizations, namely those that suffer from organizational changes tend to have complex structures. Additionally, those files tend to be modified more frequently. Due to the complex structure and high modification frequency, those files tend to have faults.

VI. Threats to Validity

A. Threats to Internal Validity

- In the analysis, files with testing code are measured and included. These files often have extreme product metric values, which may affect the analysis results.
- There are probably more factors besides those we have used in our analysis (origins, product metrics, the number

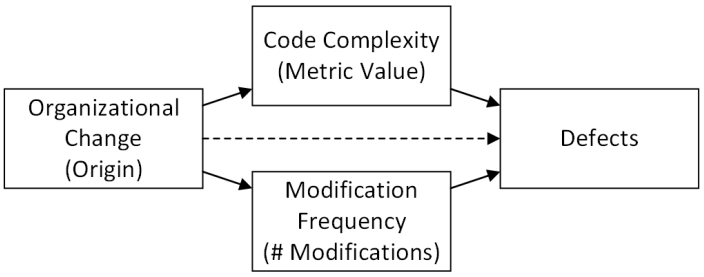


Fig. 9: Relationships between the factors

of modifications, and defects).

- The development periods of the organizations in our analysis target projects have large differences, which may significantly impact the analysis and should be considered.
- Since our approach uses diff to detect origins, code movements and renames are considered to be changes even though they are not.

B. Threats to External Validity

We conducted analysis on two open source projects, OpenOffice and VirtualBox. Because of the number of projects and their similarity (both are open source projects and are developed by Sun and Oracle), the generality is not clear. Therefore, it is desirable to add more projects in analysis. However, these projects are sufficiently practical, and significantly famous. Additionally, Sun and Oracle were involved in different stages of software development. For example, Sun is the first developer in OpenOffice, but is the second developer in VirtualBox. Therefore, these projects are appropriate projects to analysis.

VII. Conclusions

In our study, we defined the origin of a file and conducted statistical analysis using two open source software projects. We found that files found that files modified by multiple organizations or developed by later organizations tend to be faultier. The relationships between origins, product metrics, and defects indicate that these files tend to be complex, and consequently, have increased faultiness. Furthermore, the relationships between the number of modifications and other factors show that the number of modifications is related to origins and defects. Thus, the origins affect defects by changing the product metrics values and number of modifications.

These results suggest that an organizational change affects software quality, and developers need to pay attention to the quality of files modified by multiple organizations or developed by later organizations. This is especially true for developers in an organization developing software handed-down from many previous organizations..

In the future, we aim to exclude files for testing from analysis, to evaluate additional factors, and to analyze more projects with various development organizations.

References

[1] M. Conway, "How do committees invent?" *Datamation*, vol. 14, no. 4, pp. 28–31, 1968.

[2] J. O. Coplien, "A generative development-process pattern language," in *Pattern languages of program design*, J. O. Coplien and D. C. Schmidt, Eds. ACM Press/Addison-Wesley Publishing Co., 1995, pp. 183–237.

[3] A. Mockus, "Organizational volatility and its effects on software defects," in *FSE '10 Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering*, 2010, pp. 117–126.

[4] L. C. Briand, J. Wüst, J. W. Daly, and D. V. Porter, "Exploring the relationship between design measures and software quality in object-oriented systems," *J. Syst. Softw.*, vol. 51, no. 3, pp. 245–273, May 2000.

[5] T. Gyimothy, R. Ferenc, and I. Siket, "Empirical validation of object-oriented metrics on open source software for fault prediction," *IEEE Trans. Softw. Eng.*, vol. 31, no. 10, pp. 897–910, Oct. 2005.

[6] N. Nagappan, T. Ball, and A. Zeller, "Mining metrics to predict component failures," in *Proceedings of the 28th international conference on Software engineering*, ser. ICSE '06. ACM, 2006, pp. 452–461.

[7] Y. Shin, R. Bell, T. Ostrand, and E. Weyuker, "Does calling structure information improve the accuracy of fault prediction?" in *Mining Software Repositories, 2009. MSR '09. 6th IEEE International Working Conference on*, May, pp. 61–70.

[8] R. Subramanyam and M. Krishnan, "Empirical analysis of ck metrics for object-oriented design complexity: implications for software defects," *Software Engineering, IEEE Transactions on*, vol. 29, no. 4, pp. 297–310, April.

[9] S. G. Eick, T. L. Graves, A. F. Karr, J. S. Marron, and A. Mockus, "Does code decay? assessing the evidence from change management data," *IEEE Trans. Softw. Eng.*, vol. 27, no. 1, pp. 1–12, Jan. 2001.

[10] T. L. Graves, A. F. Karr, J. S. Marron, and H. Siy, "Predicting fault incidence using software change history," *IEEE Transactions on Software Engineering*, vol. 26, no. 7, pp. 653–661, July 2000.

[11] A. Hassan and R. Holt, "The top ten list: dynamic fault prediction," in *Software Maintenance, 2005. ICSM'05. Proceedings of the 21st IEEE International Conference on*, Sept. 2005, pp. 263–272.

[12] W. Lim, "Effects of reuse on quality, productivity, and economics," *Software, IEEE*, vol. 11, no. 5, pp. 23–30, Sept.

[13] N. Nagappan, B. Murphy, and V. Basili, "The influence of organizational structure on software quality: an empirical case study," in *ICSE '08 Proceedings of the 30th international conference on Software engineering*, 2008, pp. 521–530.

[14] V. R. Basili, G. Caldiera, and H. D. Rombach, "The goal question metric approach," in *Encyclopedia of Software Engineering*. John Wiley & Sons, Inc., 1994.

[15] H. Washizaki, R. Namiki, T. Fukuoka, Y. Harada, and H. Watanabe, "A framework for measuring and evaluating program source code quality," in *Proceedings of the 8th international conference on Product-Focused Software Process Improvement*, ser. PROFES '07. Springer-Verlag, 2007, pp. 284–299.

[16] Scientific Toolworks, Inc., "Understand," http://www.scitools.com/, 1996.

[17] S. Kim, T. Zimmermann, E. J. Whitehead Jr., and A. Zeller, "Predicting faults from cached history," in *Proceedings of the 29th international conference on Software Engineering*, ser. ICSE '07. IEEE Computer Society, 2007, pp. 489–498.

[18] A. Mockus and L. G. Votta, "Identifying reasons for software changes using historic databases," in *Proceedings of the International Conference on Software Maintenance (ICSM'00)*, ser. ICSM '00. IEEE Computer Society, 2000, pp. 120–.

[19] J. Śliwerski, T. Zimmermann, and A. Zeller, "When do changes induce fixes?" *SIGSOFT Softw. Eng. Notes*, vol. 30, no. 4, pp. 1–5, May 2005.