IEICE TRANS. INF. & SYST., VOL.E101–D, NO.11 NOVEMBER 2018

PAPER

Studying the Cost and Effectiveness of OSS Quality Assessment Models: An Experience Report of Fujitsu QNET

Yasutaka KAMEI[†a)], ***Member***, **Takahiro MATSUMOTO**[†], **Kazuhiro YAMASHITA**[†], ***Nonmembers***, **Naoyasu UBAYASHI**[†], ***Member***, **Takashi IWASAKI**[††], ***and*** **Shuichi TAKAYAMA**[††], ***Nonmembers***

SUMMARY Nowadays, open source software (OSS) systems are adopted by proprietary software projects. To reduce the risk of using problematic OSS systems (e.g., causing system crashes), it is important for proprietary software projects to assess OSS systems in advance. Therefore, OSS quality assessment models are studied to obtain information regarding the quality of OSS systems. Although the OSS quality assessment models are partially validated using a small number of case studies, to the best of our knowledge, there are few studies that empirically report how industrial projects actually use OSS quality assessment models in their own development process. In this study, we empirically evaluate the cost and effectiveness of OSS quality assessment models at Fujitsu Kyushu Network Technologies Limited (Fujitsu QNET). To conduct the empirical study, we collect datasets from (a) 120 OSS projects that Fujitsu QNET's projects actually used and (b) 10 problematic OSS projects that caused major problems in the projects. We find that (1) it takes average and median times of 51 and 49 minutes, respectively, to gather all assessment metrics per OSS project and (2) there is a possibility that we can filter problematic OSS systems by using the threshold derived from a pool of assessment metrics. Fujitsu QNET's developers agree that our results lead to improvements in Fujitsu QNET's OSS assessment process. We believe that our work significantly contributes to the empirical knowledge about applying OSS assessment techniques to industrial projects.

key words: *open source software, OSS quality assessment models, empirical studies, applied research*

1. Introduction

Open source software (OSS) is vital to not only end users but also software industries. Software industries can reduce their cost by applying high-quality OSS to their system development. The survey conducted by Hauge *et al.* [1] showed that close to 50% of Norwegian software industries integrate OSS components into their solutions for customers. According to a survey conducted by Black Duck [2], 78% of software industries run part or all of their business operations on OSS systems.

When software industries decide which OSS systems they integrate into their system development, they are sometimes concerned about the quality of the OSS systems and the continuity of OSS projects [3], [4]. If the quality of OSS systems is low, software industries face issues due to bugs caused by the OSS and incur unexpected costs to modify them.

Therefore, prior work has proposed OSS quality assessment models [5], [6], which provide a set of indicators for the quality of OSS systems such as their functionality and usability. For example, Petrinja *et al.* [6] showed that OSS quality assessment models provide comparable assessments for two OSS projects (i.e., the Chrome project and Firefox project) through experiments using students in bachelor's and master's programs. While studies have shown that OSS quality assessment models have the potential to be used in industrial settings, to the best of our knowledge, there are few studies that empirically report how industrial projects integrate OSS quality assessment models into their own development process.

In this paper, we set out to empirically study the cost and effectiveness of OSS assessment models in the development projects of Fujitsu QNET. Fujitsu QNET projects would like to know the risk of candidate OSS systems before integrating them into Fujitsu QNET's development. To do so, we score OSS systems on the basis of metric thresholds, similar to previous work [7], and analyze the relationship between the scores of problematic and non-problematic OSS systems. We use the internal issue tracking systems (ITSs) at Fujitsu QNET to identify problematic OSS systems. We structure our study along the following two research questions:

(RQ1) How much cost is incurred when gathering assessment metrics?
To use OSS quality assessment models, Fujitsu QNET projects need to collect assessment metrics from the OSS system that they plan to use. However, their development effort is limited. Therefore, it is important to understand how much cost is incurred when collecting assessment metrics. To evaluate RQ1, we measure the time to collect 43 assessment metrics that Fujitsu QNET developers select.

(RQ2) How effective are the OSS assessment metrics?
Fujitsu QNET projects would like to know the risk of candidate OSS systems before integrating them into Fujitsu QNET's development. To evaluate RQ2, we score OSS systems on the basis of metric thresholds, similar to previous work [7], and analyze the relationship between the scores of problematic and non-problematic OSS systems. We use the internal issue tracking systems (ITSs) at Fujitsu

Manuscript received May 10, 2018.
Manuscript revised July 6, 2018.
Manuscript publicized August 8, 2018.
[†]The authors are with Kyushu University, Fukuoka-shi, 819–0395 Japan.
[††]The authors are with Fujitsu Kyushu Network Technologies Limited, Fukuoka-shi, 814–0001 Japan.
a) E-mail: kamei@ait.kyushu-u.ac.jp
DOI: 10.1587/transinf.2018EDP7163

Table 1 Types of OSS Systems used in Fujitsu QNET development projects

Categories	Number	Percentage
Development	66	55.0
System Administration	28	23.3
Communications	12	10.0
Audio & Video	5	4.2
Home & Education	5	4.2
Security & Utilities	2	1.7
Graphics	1	0.8
Business & Enterprise	0	0
Games	0	0
Other	1	0.8
Total	120	100

QNET to identify problematic OSS systems.

The main contributions of this paper are as follows:

- An insight into OSS assessment models in industrial software projects that we derive from our quantitative and qualitative analysis.
- The thresholds of 9 OSS assessment metrics that are derived from 120 OSS projects.

This paper is an extended version of our earlier workshop paper [8]. We extend our previous work by:

- Summarizing the types of OSS systems used in Fujitsu QNET development projects for better understanding the context of our empirical study (Table 1).
- Presenting results of the time that we spend to measure OSS assessment metrics with respect to each assessment metric (RQ1).
- Conducting interviews with Fujitsu QNET's senior developers about our findings in each RQ1 and RQ2.
- Adding a threats to validity section to show limitations of our work.

Paper organization. The remainder of the paper is organized as follows. Section 2 introduces the background of our study. Section 3 explains the OSS usage and assessment at Fujitsu QNET. Section 4 presents our discussion with the stakeholders at Fujitsu QNET. Sections 5 and 6 present the results. Section 7 discloses the threats to the validity of our findings. Section 8 draws the conclusions of this study.

2. Background and Related Work

Previous studies have proposed OSS quality assessment models such as the Open Source Maturity Model (OSMM) [9], the Open Business Readiness Rating (OpenBRR) [10], Qualification and Selection of Open Source software (QSOS) [11], Open BQR [12], OpenSource Maturity Model (OMM) [13], and RepOSS [5]†. Briefly speaking, OSS quality assessment models provide (1) a set of indicators for the quality of OSS systems, such as the functionality and usability, and (2) a process of scoring OSS systems based on the indicators.

There are several studies that have evaluated OSS assessment models [6], [14], [15]. For example, Deprez and Alexandre [14] compared QSOS and OpenBRR on the basis of description of the methodologies (i.e., no empirical evaluation) according to their scoring procedures and evaluation criteria. Petrinja *et al.* [6] conducted experiments to empirically evaluate three OSS assessment models (OpenBRR, QSOS, and OMM). In their experiments, bachelor's and master's students assessed two OSS systems as research participants (Firefox and Chrome) using the three OSS assessment models. The results showed that the mean scores derived from the research participants are quite similar among the three models. Petrinja and Succi [15] also assessed six OSS systems using the OMM. The study assessment results by the authors and master's students demonstrated the applicability of the OMM.

Although previous studies have shown that the OSS quality assessment models have the potential to be used in industrial settings, they do not evaluate whether or not the OSS systems assessed as risky actually caused problems in industrial projects. Therefore, we report the empirical studies on OSS quality assessment models at Fujitsu QNET according to effectiveness.

3. OSS Usage and Assessment at Fujitsu QNET

Fujitsu QNET generally develops network systems and embedded systems. We target 120 OSS systems that the Fujitsu QNET's development projects used for two years (2015–2016).

Table 1 summarizes the types of OSS systems used in Fujitsu QNET development projects. We sort OSS categories based on the percentages of OSS systems used in Fujitsu QNET development projects in descending order. We manually classify OSS systems into the categories that are used in SourceForge††. For example, we classify the OSS systems about network monitoring tools and logging tools into "Communications" and "Development." The top three categories of OSS systems used at Fujitsu QNET are "Development," "System Administration," and "Communications," since network systems and embedded systems are mainly developed at Fujitsu QNET.

OSS assessment models. Overall, OSS quality assessment models provide (1) a set of indicators for the quality of OSS systems and (2) a process of scoring OSS systems based on the indicators [12]. Figure 1 shows the process of how Fujitsu QNET will make use of OSS assessment models in their development projects.

Step 1. Developers at Fujitsu QNET collect OSS assessment metrics for the OSS system (a target OSS system) that they want to assess (Details in Sect. 5).

Step 2. Thresholds for the OSS assessment metrics are derived from the datasets that are collected from a pool of 120 OSS systems (Sect. 6).

Step 3. Developers score the OSS system on the basis of

†The authors show only an abbreviation for BQR and RepOSS.

††https://sourceforge.net/

Table 2 Expected Quality of OSS systems for Each Project Team

#	Purpose	Knowledge level	Comments about Expected Quality for OSS systems
1	R&D	Black Box	*"We often use the OSS systems that implement emerging technologies because our team develops experimental products. In addition, such OSS systems are likely to not be matured yet. Therefore, we expect that we can easily trace and manage the status of issues for the OSS systems."*
2		Black Box	*"Since experimental products need to be developed in a short period, it is important to easily use OSS systems (easy to install and run them)."*
3		Black Box	*"We expect that there are many references online to see how to use OSS systems and how to solve the problems we may face in future."*
4		Black Box	*"When we finish developing this experimental product, we shift this project to release the product for market. Therefore, we need to easily set up an environment for conducting performance testing for OSS systems."*
5		Black Box	*"For experimental products, we expect that OSS systems have emerging technologies and can be executed at least in a default scenario. We do not expect that they can be executed under expectational or error conditions."*
6		Black Box	*"Because our development period is short, we want to use the OSS systems of which sample source code is fully presented."*
7	Products for market	White Box	*"We would like to use mature OSS systems. Therefore, a considerable period has elapsed since the OSS systems were released."*
8		Black Box	*"The quality of OSS systems must be high. We usually check how often the OSS systems are used for commercial purposes."*

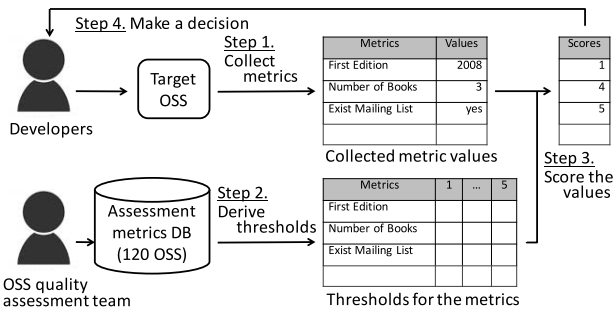


Fig. 1 The approach of OSS quality assessment

the thresholds. (Sect. 6).

Step 4. Developers make a decision about whether or not they apply it to their project (Sect. 6).

4. Preliminary Survey

4.1 Overview

Each development team at Fujitsu QNET may have different contexts for using OSS systems. To better understand how Fujitsu QNET's developers use OSS systems, we conducted semi-structured interviews at Fujitsu QNET.

4.2 Approach

Semi-structured interviews begin with a set of prepared questions, but the structure of the interview is flexible, allowing the interviewer (one of the authors) to further query unexpected answers from the interviewees by developing new questions during the session [16], [17]. Many of the questions are directly connected with our research interests, such as: "How does your project team use OSS systems?" or "What does your project team expect for OSS systems?" All of the conversations were recorded and coded by the interviewer.

In this study, we conducted semi-structured interviews of eight project teams at Fujitsu QNET. One project manager (more than 10 years experience) and one or two senior developers (more than 6 years experience) from each project team participated the interview. We spent about 20–30 minutes per project team.

4.3 Results

Table 2 summarizes the results of our interview. Although six out of eight project teams use OSS systems for R&D[†], the other two teams use them for products for market.

If Fujitsu QNET's teams only make an effort to learn how to use OSS systems, especially their inputs and outputs without knowledge of their internal operation, we use "Black box" in the column "Knowledge level." If they make an effort to analyze source code files and obtain knowledge of their internal operation, we use "White box" in the column. We find that OSS systems are likely to be used as "Black box," since one of the major purposes of using OSS systems is to reduce development cost. However, one of the two teams that develop their products for market uses OSS systems as "White box" because it needs to modify source code when it faces problems caused by OSS systems.

We find that there are several common expected qualities for OSS systems in Table 2. For example, we can see that one common expected quality is easy-to-use for team #2, #3, and #6. We also find that they want to apply OSS systems to their team (#1 and #5) for emerging technologies. On the other hand, for the teams that develop products for market, they value mature OSS systems, as evidenced by "a considerable period has elapsed since OSS systems were released" and "we check how many times the OSS systems are used for commercial purposes" (#7 and #8).

[†]We use R&D as the context of developing trial and/ experimental products and not products for market.

At Fujitsu QNET, when OSS systems are used for R&D, developers are likely to use OSS systems that have emerging technologies and are easy-to-use. On the other hand, when OSS systems are used for products for market, developers are likely to value the maturity of the OSS systems.

5. (RQ1) How Much Cost is incurred when Gathering Assessment Metrics?

5.1 Overview

To conduct the OSS assessment, Fujitsu QNET needs to collect assessment metrics in two cases. First, when developers assess the OSS system that they want to integrate into their project, they collect assessment metrics from the OSS system. Second, when OSS systems are continuously developed and the metric values in the OSS assessment repositories become obsolete, maintainers need to re-collect metrics for the OSS systems to update the metric values. The frequency would depend on each OSS system because the speed of software development varies among OSS systems.

It is important to understand the cost (i.e., how long it takes to collect assessment metrics), since the development effort is limited in industrial projects. If the cost is not negligible, it is difficult to apply OSS assessment techniques to industrial projects. Therefore, for this RQ, we investigate the costs of collecting assessment metrics per OSS system. This RQ is related to step 1 in Fig. 1.

5.2 Approach

To answer this RQ, we conduct an experiment with three graduate students in computer science as research participants. Each participant searches and collects assessment metrics online and records the time that he/she spends for each assessment metric. Although this experiment requires a large amount of effort (255 hours in total), we need to conduct additional experiments with more participants to mitigate the impact of individual differences on our results. We elaborate on the weakness of our current experimental design in Sect. 7.

We asked research participants to follow the documents we composed in order to prevent differences in the collection of assessment metrics among the participants. In the case where they could not find any information about an arbitrary metric until 15 minutes, we asked them to stop searching the metric and record the metric value as null. We do not include time for creating scripts in the time for spending to measure OSS assessment metrics, because we would like to evaluate cost in the situation where industrial projects have such scripts and re-collect metrics values (OSS systems are often continuously developed). Note that, to create scripts, we spent about 90 minutes for analyzing log messages that are collected from version control systems and 90 minutes for analyzing issue tracking systems.

Target OSS projects. In this survey, we selected 120 OSS projects based on the OSS usage at Fujitsu QNET.

Target assessment metrics. In this study, we list the assessment metrics that are defined in OSMM [9], OpenBRR [10], and RepOSS [5]. Then, we had a discussion with Fujitsu QNET's developers to select 43 assessment metrics[†] in five dimensions when considering the results of Sect. 4. We selected the 43 assessment metrics because our resource is limited and not enough to target all assessment metrics for our experiments. The left four columns in Table 3 summarizes a few of metrics in each dimension as an example. Unfortunately, we cannot disclose the list of all 43 metrics we selected for confidentiality reasons.

We now describe each dimension and its assessment metrics in more detail.

Activeness dimension: Not all OSS projects are always active [29]–[31]. Khondhu *et al.* [31] showed that only 6% of OSS projects maintain their activity. The use of OSS provided by inactive projects is risky, since the provided OSS may not be updated and become obsolete in future.

To calculate the average number of commits per month (i.e., ANC) and the percentage of commits from the developer who has the largest number of commits in the project (i.e., PCD), we use scripts to analyze log messages that are collected from version control systems (VCSs) of the target OSS project.

We collect issue reports from issue tracking systems (ITSs) such as Bugzilla and Jira and use scripts to analyze them to calculate the average number of days for bug fixes (i.e., NDAY). We find the year of the first release (i.e., to measure REL) from the release notes of the OSS's website and Wikipedia or the release tags of GitHub repositories.

Community dimension: OSS development is often supported by volunteer developers [32], [33]. The community of users and developers submits various types of contributions (e.g., modifications, bug fixes, and bug reports) to the OSS project. Using OSS provided by projects that are less mature is risky, since the provided OSS may have less discussion for development and be lower quality.

Similar to ANC and PCD, we use scripts to analyze the log messages of VCSs to calculate the number of core developers (i.e., NC). Inspired by previous studies [22]–[24], [34], we use heuristics that define the core developers as those who produce roughly 80% of the total contributions. To obtain ND and NT, we find the developer and user mailing lists from the menus (e.g., mailing lists and discussion) of OSS websites.

Documentation dimension: Software documentation is important for users. It explains how OSS operates and/or how to be used. Using OSS provided by projects that have less documentation is risky, since Fujitsu QNET's developers cannot find the solution for problems that are caused by OSS systems during their development phase.

We search for the name of the OSS in the category

[†]Our earlier workshop paper [8] showed 44 assessment metrics because we double-counted one of the assessment metrics to show the number of assessment metrics.

Table 3 Summary of OSS assessment metrics

Dim.	Name	Definition	Rationale	Pre-Condition			Time (minutes)	
				C1	C2	C3	Ave.	Med.
Activeness	ANC	Average number of commits per month	OSS systems that have many commits are maintained well and less likely to be defect-prone.	✓	✓		0.5	0.5
	PCD	Percentage of commits from the developer who has the largest number of commits	If OSS systems are more likely to depend on one developer, they are likely to be terminated, since the developer is a bus factor of the project [18]–[20].	✓	✓		0.5	0.5
	NDAY	Average number of days for bug fixes	OSS systems that have a lower average number of days for bug fixes are likely to fix a new reported bug.	✓		✓	0.5	0.5
	REL	The year of the first release	An OSS system with a longer history is the more likely to have defects that are solved [21].	✓	✓		2.4	2.0
Community	NC	Number of core developers	A higher number of core developers means that it is more likely that defects are solved, since the core developers have many contributions to OSS systems [22]–[24].	✓	✓		0.5	0.5
	ND	Number of threads in developer mailing lists	More discussion results in a higher chance of high-quality OSS systems [25], [26].	✓			2.9	2.0
	NT	Number of threads in user mailing lists	More discussion with users means that it is more likely that an OSS community might be supportive of users and be good to use [25].	✓			2.1	2.0
Documentation	NB	Number of books	More books mean that there is a higher chance of available information when Fujitsu QNET developers face the problem in use [27].				1.1	1.0
	NTA	Number of technical articles	More online technical articles mean that Fujitsu QNET developers are more likely to solve the problem that they will face in use [25], [27].				1.1	1.0
	NP	Number of technical presentations	More technical presentations mean that it is more likely that OSS systems might provide useful information to solve the problem that Fujitsu QNET developers face in use [27].				1.2	1.0
Reliability	NBUG	Number of reported bugs	OSS systems that have more reported bugs are less likely to have unknown bugs [25].	✓		✓	0.5	0.5
	PFIX	Percentage of fixed bugs in all reported bugs	OSS systems that have more fixed bugs are less likely to have unknown bugs.	✓		✓	0.5	0.5
Portability	MUL	Multilingualization	The OSS that supports multilingualization is likely to be easy when Fujitsu QNET developers support non-English language in their use.	✓			1.2	1.0
	OS	Supported OS	There may be some specific OS that introduces more defects for the support [28].	✓			1.6	1.0
	INS	Exist of a installer package	A installer package helps Fujitsu QNET developers to install the OSS.	✓			1.4	1.0

C1: OSS systems' websites are found in advance C2: OSS systems' VCSs are found in advance C3: OSS systems' ITSs are found in advance

"Books" of Amazon and use the number of results as the number of books (i.e., NB). Similarly, we search for the name of the OSS on technical websites that Fujitsu QNET's developers choose[†] as the number of online articles (i.e., NA) and in the category "Technology" of SlideShare as the number of technical presentations (i.e., NP).

Reliability dimension: OSS continuously evolves to fix a bug or add a new feature [35], [36]. Using OSS provided by projects that are less positive for fixing bugs is risky, since Fujitsu QNET's developers may face problems that are caused by bugs that are not fixed nor reported yet [25].

Similar to NDAY, we collect issue reports from ITSs and use scripts to analyze them to calculate the number of reported bugs (i.e., NBUG) and the percentage of fixed bugs of all reported bugs (i.e., PFIX).

Portability dimension: Fujitsu QNET's products may be released to multiple countries and on multiple platforms. Therefore, using OSS that is less portable is risky, since Fujitsu QNET's developers may spend a large amount of effort to address customers' requirements such as multiple platforms in the future.

We find the information about whether or not an OSS system supports multilingualization (i.e., MUL), which OSs are supported (i.e., OS), and whether or not it provides an installer package from the release notes of the OSS's website and Wikipedia or the readme page of GitHub repositories.

5.3 Results

The right part of Table 3 shows the results of the time that we spent to measure OSS assessment metrics. C1 in the Precondition column shows that, in advance, we find the websites that are managed by the OSS community. Therefore, we do not include the time to find the websites when we collect the metrics that has C1 as a precondition. C2 and C3 show that, in advance, we find the VCSs and ITSs of OSS projects. The time is also not included to find VCSs and ITSs. The time that we spend for C1, C2, and C3 are 1, 2, and 2 minutes, respectively, for each precondition.

Table 3 indicates that "the number of threads in de-

[†]e.g., https://www.infoq.com/

veloper mailing lists" is the most costly evaluation metric according to the time spent and is 3 minutes on average. Compared with the VCSs and ITSs, OSS projects are likely to put their mailing lists in a number of different forms for each OSS. Therefore, it takes more time to determine the structure of the data of mailing lists.

To assess three preconditions and collect 43 evaluation metrics, it takes 27 minutes at minimum, 51 minutes on average, a median of 49 minutes, and 86 minutes at maximum. There is almost 1 hour difference between the minimum and maximum values (86 - 27 minutes). The main reason is the structure of the websites of each OSS project. If OSS projects provide information related to assessment metrics across multiple pages, it takes much more time to find all of the information.

Feedback from Fujitsu QNET developers. We conducted interviews with Fujitsu QNET's three senior developers. The purpose of the interview is to understand whether or not it is acceptable to spend a median time of 49 minutes in practical settings. Our questions are: (1) Can you accept a time of less than 1 hour per OSS?, (2) Are there any comments that we should improve? The interview time is about 30 minutes.

According to Fujitsu QNET, all three senior developers can accept a time of less than 1 hour per OSS project. Therefore, it is reasonable for Fujitsu QNET to apply an OSS assessment technique from the viewpoint of cost.

However, three senior developers also point out that it is desirable for industries to reduce costs as much as possible. Furthermore, there are several OSS projects that require more than 1 hour to assess three preconditions and collect 43 evaluation metrics. In the future, we will study an approach that semi-automatically collects assessment metrics using natural language processing techniques.

It takes average and median times of 51 and 49 minutes, respectively, to collect all assessment metrics per OSS system.

6. (RQ2) How Effective are OSS Assessment Metrics?

6.1 Overview

To elevate the use of OSS assessment metrics from measurement to decision-making, it is essential to derive meaningful threshold values. Similar to previous work [7], [37], we use an approach that empirically derive the metric threshold values from the measurement data of a benchmark in Sect. 5.

To evaluate the effectiveness of the OSS assessment techniques, we classify OSS systems on basis of threshold values into one of five risk groups. Then, we see how the OSS systems in each risk group are included in two groups: (1) OSS systems that cause major problems† during Fujitsu QNET's development and (2) OSS systems that do not cause problems. Fujitsu QNET records reported problems in their internal ITSs. Although the part in which the threshold values are derived is related to step 2 in Fig. 1, the part in which OSS systems are compared is related to step 3.

Target OSS projects. We use the same 120 OSS projects mentioned in Sect. 5. We assume that they are non-problematic OSS systems, since any problems are not reported in Fujitsu QNET's internal ITSs. We also use 10 OSS projects that are reported as OSS systems that cause major problems in the ITSs. In summary, we use 120 OSS projects to derive threshold values and score 130 OSS projects to validate the effectiveness of the threshold values.

Target assessment metrics. In this study, we use 16 out of the 43 assessment metrics that have an interval or ratio scale, since the categorical variables (e.g., MUL and OS in Table 3) cannot provide threshold values.

6.2 Approach

Deriving threshold values. We determine threshold values on the basis of the distribution of the assessment metrics from the 120 OSS projects in Sect. 5, similar to Alves *et al.* [7].

For each assessment metric, we sort the metric values of the OSS projects in descending (or ascending) order. Then, we use the thresholds derived by choosing 10%, 20%, 30%, and 40% of the overall OSS projects. Furthermore, these percentiles are used to characterize the OSS according to five categories: very low risk ($\leq$ 10%), low risk (10–20%), moderate risk (20–30%), high risk (30–40%) and very high risk ($\geq$ 40%).

We use an example to illustrate how we derived threshold values. For example, the values of NC for 10 OSS projects are 1, 3, 5, 7, 9, 11, 13, 15, 17, and 19. We obtain 19, 17, 15, 11, 9, 7, 5, 3, 1 by sorting the metrics values in descending order. The 1st threshold is $\geq$ 19, the 2nd threshold is $\geq$ 17, the 3rd threshold is $\geq$ 15, and the 4th threshold is $\geq$ 13. Therefore, given two new projects A and B that Fujitsu QNET's developers evaluate, if the values of NC are 16 and 12, project A is classified in the moderate risk group because it is between 15 (3rd threshold) and 17 (2nd threshold). On the other hand, project B is classified in the very high risk group because it is less than 13 (4th threshold).

Scoring OSS systems. When we use the technique for deriving the threshold values, we can estimate the risk of OSS systems according to one assessment metric. To comprehensively assess OSS systems using a set of assessment metrics, we combine the results of the assessment metrics into a single output.

We decided to use a simple approach to score OSS systems on the basis of Table 4, since it would help us to communicate our results with Fujitsu QNET's developers. We give a rating of 5 to an OSS system in each assessment metric if the OSS system is classified in the very low risk group (the risk group A). Similarly, we give ratings 1, 2, 3, or 4 to an OSS system if it is classified in the very high (E), high

†The examples of major problems are: (a) An OSS system has bugs that do not release memory back to an operating system and (b) Fujitsu QNET's projects notice that an OSS system actually does not provide the APIs that are described on its website.

Table 4 Thresholds of OSS Assessment Metrics

Dim.	Metrics	Order	Threshold between two risk groups			
			E - D	D - C	C - B	B - A
Act.	ANC	Desc	43	74	120	158
	PCD	Asc	0.27	0.22	0.14	0.08
	NDAY	Asc	137	68	49	11
	REL	Asc	2005	2002	2000	1996
Com.	NC	Desc	125	176	313	688
	ND	Desc	1,719	4,593	12,236	29,519
	NT	Desc	705	1,112	1,439	13,700
Doc.	NB	Desc	0	2	3	6
	NTA	Desc	749	1,380	3,080	13,700
	NP	Desc	695	1,069	6,099	124,256
Rel.	NBUG	Desc	2,224	3,236	5,436	11,578
	PFIX	Desc	0.87	0.92	0.94	0.96

E: very high risk group A: very low risk group

(D), moderate (C), or low risk group (B), respectively. We calculate the average value of the scores to obtain a single output.

To understand how well the OSS assessment metrics work, we compare the outputs of two groups of OSS with/without problems. Although this study evaluates the research topic that accelerates academic–industrial collaboration, it also has the weakness of our current approach. We discuss the weakness of our current approach in Sect. 7.

6.3 Results

Deriving threshold values. Table 4 shows the threshold values of each assessment metric. For example, if the project has 1994 as the year of the first release (REL), the project is classified in the very low risk group from the aspect of REL.

We found one problem in our derived threshold values. In NP, there is a difference of two orders of magnitude (124,256 vs. 6,099) between the threshold values of the very low and low risk groups.

To understand whether or not 124,256 is reasonable number, we search "apache," which is one of the most well-known OSS systems, in the field of SlideShare. Even if we search the well-known OSS system, there are only 23,976 slides (= 12 slides per page × 1,998 pages)[†]. One possible reason is that 124,256 includes slides that are not related to OSS systems because the names of OSS systems sometimes have general terms. We also find a similar problem with the other metrics (NB: number of books and NTA: number of technical articles) in the Documentation dimension. We decided to remove NB, NTA, and NP from our assessment metrics. We need to improve the steps to obtain those three metrics in future.

Scoring OSS systems. Table 5 shows the percentages of problematic and non-problematic OSS systems for each score. We find that all OSS systems over 4.0 are non-problematic. Furthermore, the score of only one problematic OSS system is more than 3.0. Therefore, the scoring approach has the potential to assess OSS systems by filtering OSS systems that have a score less than 3.

[†]http://www.slideshare.net/search/slideshow?searchfrom=header&q=apache

Table 5 Distribution of the scores for non-problematic and problematic OSS systems

Score	% of non-problematic OSS systems	% of problematic OSS systems
≤ 2	22.7	30.0
≤ 3	47.3	60.0
≤ 4	24.5	10.0
> 4	5.5	0.0

A higher score indicates that the OSS systems are higher quality

Table 6 Scoring results of problematic OSS systems in each metric

Dim.	Metrics	Statistic			# systems having each score				
		Ave.	Var.	Med.	5	4	3	2	1
Act.	ANC	3.3	1.81	3.0	2	2	3	2	1
	PCD	1.8	1.96	1.0	1	1	0	1	7
	NDAY	1.0	0	1.0	0	0	0	0	10
	REL	2.5	2.65	2.0	2	1	2	0	5
Com.	NC	2.6	3.24	1.5	3	1	0	1	5
	ND	2.3	2.61	1.5	3	0	0	2	5
	NT	2.9	2.89	3.0	3	1	2	0	4
Rel.	NBUG	2.7	2.41	2.0	2	2	0	3	3
	PFIX	2.3	2.21	2.0	2	0	0	4	4

Score 5 is given to OSS systems that are categorized as the very-low risk group. Ave.: Average, Var.: Variance, Med.: Median

To better understand which metrics are likely to provide a lower score for problematic OSS systems, Table 6 shows the average and median values of each metric for problematic OSS systems and the number of them for each score. We find that NDAY provides a score of 1 for all problematic OSS systems. On the other hand, for ANC, there is at least one OSS system having each score.

Feedback from Fujitsu QNET developers. We have consulted with Fujitsu QNET's senior developers for about two hours. The purpose of the interview is to arrive at a deeper understanding of our results (e.g., how well does our approach work for Fujitsu QNET's development? and how can we improve our current approach?).

We find that OSS systems with a score greater than 4 are not problematic. Fujitsu QNET's developers agree that our results lead to improvements in Fujitsu QNET's OSS assessment process because of high precision. Fujitsu QNET's project teams expect that they would save the effort that they usually spend to assess OSS systems while keeping assessment quality[††]. Therefore, they decided to integrate the approach into their actual development process. One way to integrate the approach is that they would allocate only a certain percentage of the effort to OSS systems with a score greater than 4 and 3, respectively (e.g., 50% and 80%).

Our scoring approach is derived using only 120 OSS systems. Therefore, to improve the validity of the experimental results, Fujitsu QNET developers want us to continu-

[††]The examples of assessing OSS systems are: how an OSS system performs on a specific hardware and whether or not the OSS system provides the APIs that the projects would like to use.

ously evaluate the approach using the OSS projects obtained during actual integration.

Fujitsu QNET's developers also would like to improve the performance of our scoring approach, since Table 4 indicates that the score of one problematic OSS system is more than 3. Their feedback for improving the performance is that we weight the assessment metrics to score OSS systems on the basis of the Fujitsu QNET's developers' knowledge. For example, project teams with "emerging technologies" may have different context of "the year of the first release" (REL) from project teams implementing infrastructure. We will conduct a developer survey of their knowledge in the future.

By using OSS systems with a score greater than 4.0 based on a threshold approach, there is a possibility that we can avoid using problematic OSS systems.

7. Threats to Validity

Construct Validity: considers the relationship between theory and observation, in case the measured variables do not measure the actual factors. We score OSS systems by calculating the average of the values derived from threshold approaches [7]. Although it is simple and intuitive to score them, in certain cases, this score may fully measure the risk of OSS systems. As we discuss in Sect. 6, we may need to consider weighting the assessment metrics based on the developers' knowledge. That said, our initial results in Sect. 6 show that the simple approach highlights non-problematic OSS systems by eliminating candidate OSS systems that have a score less than 4.

External Validity: considers the generalization of our findings. Our study focuses on the usage of OSS systems in one software industry. Therefore, this project may not be representative of the usage of OSS systems in all software industries. However, since we show the impact of research methodologies on industrial settings, we believe that our work significantly contributes to empirical knowledge of the application of OSS assessment techniques to industrial projects.

Although we choose 43 OSS assessment metrics, there may be other features that we did not measure for the OSS assessment metrics. Although we can collect all 43 metrics within 1 hour (i.e., RQ1 shows that it takes a median time of 49 minutes per OSS system), further studies using other metrics may improve the performance of detecting problematic OSS systems.

Internal Validity: refers to whether the experimental conditions make a difference or not, and whether there is sufficient evidence to support the claim being made. We used Fujitsu QNET' ITSs to identify which OSS systems cause major problems. Although Fujitsu QNET has long development history using OSS systems, there may be other major problems that have not occurred yet at Fujitsu QNET and are not archived in the ITSs.

To conduct RQ1, we used three graduate students in computer science. Although they are in the last year of their program, developers for industrial projects can more quickly collect OSS assessment metrics than students.

8. Conclusion

In this paper, we empirically evaluated the cost and effectiveness of OSS quality assessment models. To conduct the empirical study, we collected the datasets from 120 OSS projects used in Fujitsu QNET's projects and 10 problematic OSS projects that caused major problems in the projects. We find that (1) average and median times of 51 and 49 minutes, respectively, are required to collect all assessment metrics per OSS project and (2) there is a possibility that we can avoid using problematic OSS systems by filtering the OSS systems with a score less than 4.0 based on a threshold approach.

We also conducted interviews with Fujitsu QNET's developers to qualitatively evaluate our findings and obtain their feedback to improve our approach. Overall, their feedback is positive; (1) it is acceptable to spend a median time of 49 minutes and (2) our approach has the potential to detect problematic OSS systems in Fujitsu QNET's context. Currently, OSS quality assessment models have been integrated into Fujitsu QNET's development projects (i.e., they collect metrics, score OSS systems, and make a decision).

The plans for future work are as follows:

- We will improve the collection of OSS assessment metrics and introduce automation to reduce costs.
- We will improve our scoring approach by weighting the assessment metrics based on the Fujitsu QNET's developers' knowledge.

References

[1] Ø. Hauge, C.F. Sørensen, and R. Conradi, "Adoption of open source in the software industry," Proc. Int'l Conf. on Open Source Systems (OSS), pp.211–221, 2008.
[2] B. Duck, "The tenth annual future of open source survey," 2016. https://www.blackducksoftware.com/future-of-open-source
[3] R. Ferenc, I. Siket, and T. Gyimothy, "Extracting facts from open source software," Proc. Int'l Conf. Software Maintenance (ICSM), pp.60–69, Sept. 2004.
[4] U. Raja and M. Tretter, "Defining and evaluating a measure of open source project survivability," IEEE Trans. Software Engineering, vol.38, no.1, pp.163–174, 2012.
[5] O. Northeast Asia, "RepOSS: A flexible OSS assessment repository," 2012.
[6] E. Petrinja, A. Sillitti, and G. Succi, "Comparing OpenBRR, QSOS, and OMM assessment models," Proc. Int'l Conf. Open Source Systems (OSS), vol.319, pp.224–238, 2010.
[7] T.L. Alves, C. Ypma, and J. Visser, "Deriving metric thresholds from benchmark data," Proc. Int'l Conf. Software Maintenance (ICSM), pp.1–10, 2010.
[8] T. Matsumoto, K. Yamashita, Y. Kamei, N. Ubayashi, T. Iwasaki, and S. Takayama, "A survey for establishing preliminary evaluation technique for open source software (in japanese)," Proc. Japanese Domestic Workshop on Foundation of Software Engineering (FOSE), pp.3–12, 2016.
[9] F.W. Duijnhouwer and C. Widdows, "Open source maturity model,"

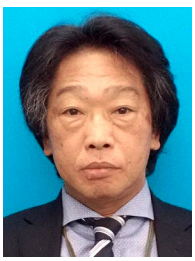
in Capgemini Expert Letter, 2003. https://jose-manuel.me/thesis/references/GB_Expert_Letter_Open_Source_Maturity_Model_1.5.3.pdf
[10] T. Wasserman and A. Das, "Using flossmole data in determining business readiness ratings," Proc. Workshop on Public Data about Software Development (WoPDaSD), pp.1–6, 2007.
[11] Atos, "Qualification and selection of open source software (QSOS), version 2.0," 2013. http://backend.qsos.org/download/qsos-2.0_en.pdf
[12] D. Taibi, L. Lavazza, and S. Morasca, "OpenBQR: a framework for the assessment of OSS," Proc. Int'l Conf. Open Source Systems (OSS), pp.173–186, 2007.
[13] E. Petrinja, R. Nambakam, and A. Sillitti, "Introducing the open-source maturity model," Proc. Int'l Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development, pp.37–41, 2009.
[14] J.C. Deprez and S. Alexandre, "Comparing assessment methodologies for free/open source software: OpenBRR and QSOS," Proc. Int'l Conf. Product-Focused Software Process Improvement (PROFES), pp.189–203, 2008.
[15] E. Petrinja and G. Succi, "Assessing the open source development processes using OMM," Advances in Software Engineering, vol.2012, pp.1–17, 2012.
[16] J. Shimagaki, Y. Kamei, S. McIntosh, A.E. Hassan, and N. Ubayashi, "A study of the quality-impacting practices of modern code review at sony mobile," Proc. Int'l Conf. Software Engineering, SEIP Track, ICSE '16, New York, NY, USA, pp.212–221, ACM, 2016.
[17] C.B. Seaman, "Qualitative methods," Guide to Advanced Empirical Software Engineering, pp.35–62, 2008.
[18] F. Ricca, A. Marchetto, and M. Torchiano, "On the difficulty of computing the truck factor," Proc. Int'l Conf. Product-Focused Software Process Improvement (PROFES), vol.6759, pp.337–351, 2011.
[19] M. Torchiano, F. Ricca, and A. Marchetto, "Is my project's truck factor low?: Theoretical and empirical considerations about the truck factor threshold," Proc. Int'l Workshop on Emerging Trends in Software Metrics (WETSoM), pp.12–18, 2011.
[20] G. Avelino, M.T. Valente, and A. Hora, "What is the truck factor of popular github applications? a first assessment," 2015. https://doi.org/10.7287/peerj.preprints.1233v2
[21] J.W. Paulson, G. Succi, and A. Eberlein, "An empirical study of open-source and closed-source software products," IEEE Trans. Software Engineering, vol.30, no.4, pp.246–256, 2004.
[22] M. Goeminne and T. Mens, "Evidence for the pareto principle in open source software activity," Joint Proc. 1st Int'l Workshop on Model Driven Software Maintenance and 5th Int'l Workshop on Software Quality and Maintainability, pp.74–82, 2011.
[23] A. Mockus, R.T. Fielding, and J.D. Herbsleb, "Two case studies of open source software development: Apache and mozilla," ACM Trans. Software Engineering and Methodology, vol.11, no.3, pp.309–346, 2002.
[24] K. Yamashita, S. McIntosh, Y. Kamei, A.E. Hassan, and N. Ubayashi, "Revisiting the applicability of the pareto principle to core development teams in open source software projects," Proc. Int'l Workshop on Principles of Software Evolution (IWPSE), pp.46–55, 2015.
[25] D. Spinellis, G. Gousios, V. Karakoidas, P. Louridas, P.J. Adams, I. Samoladas, and I. Stamelos, "Evaluating the quality of open source software," Electronic Notes in Theoretical Computer Science, vol.233, pp.5–28, 2009.
[26] S. McIntosh, Y. Kamei, B. Adams, and A.E. Hassan, "An empirical study of the impact of modern code review practices on software quality," Empirical Software Engineering, vol.21, no.5, pp.2146–2189, 2016.
[27] M. Sarrab and O.M.H. Rehman, "Empirical study of open source software selection for adoption, based on software quality characteristics," Advances in Engineering Software, vol.69, pp.1–11, 2014.
[28] M. Michlmayr, "Software process maturity and the success of free software projects," Proc. Conf. Software Engineering: Evolution and Emerging Technologies, pp.3–14, 2005.
[29] S. Krishnamurthy, "Cave or community? an empirical examination of 100 mature open source projects," First Monday, vol.7, no.6, 2002.
[30] R. English and C.M. Schweik, "Identifying success and tragedy of floss commons: A preliminary classification of sourceforge.net projects," Proc. Int'l Workshop on Emerging Trends in FLOSS Research and Development (FLOSS), pp.54–59, 2007.
[31] J. Khondhu, A. Capiluppi, and K.-J. Stol, "Is it all lost? a study of inactive open source projects," Proc. Int'l Conf. Open Source Systems (OSS), vol.404, pp.61–79, 2013.
[32] E.S. Raymond, The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary, 1999.
[33] A. Senyard and M. Michlmayr, "How to have a successful free software project," Proc. Asia-Pacific Software Engineering Conference (APSEC), pp.84–91, 2004.
[34] G. Robles, S. Koch, J.M. González-Barahona, and J. Carlos, "Remote analysis and measurement of libre software systems by means of the cvsanaly tool," Proc. Int'l Workshop on Remote Analysis and Measurement of Software Systems (RAMSS), pp.51–55, 2004.
[35] M.W. Godfrey and Q. Tu, "Evolution in open source software: A case study," Proc. Int'l Conf. Software Maintenance (ICSM), pp.131–142, 2000.
[36] G. Robles, J.M. Gonzalez-barahona, G.D.S.Y. Comunicaciones, and M. Michlmayr, "Evolution of volunteer participation in libre software projects: evidence from debian," Proc. Int'l Conf. Open Source Systems (OSS), pp.100–107, 2005.
[37] K. Yamashita, C. Huang, M. Nagappan, Y. Kamei, A. Mockus, A.E. Hassan, and N. Ubayashi, "Thresholds for size and complexity metrics: A case study from the perspective of defect density," Proc. Int'l Conf. Software Quality, Reliability and Security (QRS), pp.191–201, 2016.

Yasutaka Kamei is an associate professor at Kyushu University in Japan. He has been a research fellow of the JSPS (PD) from July 2009 to March 2010. From April 2010 to March 2011, he was a postdoctoral fellow at Queen's University in Canada. He received his B.E. degree in Informatics from Kansai University, and the M.E. degree and Ph.D. degree in Information Science from Nara Institute of Science and Technology. His research interests include empirical software engineering and Mining Software Repositories (MSR).

Takahiro Matsumoto received his Bachelor's degree and Master's degree from Kyushu University. His research interests include software engineering, data mining, mining software repositories (MSR).

Kazuhiro Yamashita received his Bachelor's degree, Master's degree, and Ph.D. degree from Kyushu University. His research interests include software engineering, data mining, mining software repositories (MSR).

Naoyasu Ubayashi is a professor at Kyushu University since 2010. He is leading the POSL (Principles of Software Languages) research group at Kyushu University. Before joining Kyushu University, he worked for Toshiba Corporation and Kyushu Institute of Technology. He received his Ph.D. from the University of Tokyo. He is a member of ACM SIGPLAN, IEEE Computer Society, and Information Processing Society of Japan (IPSJ). He received IPSJ SIG Research Award 2003.



Takashi Iwasaki graduated Kagoshima University in 1985. In the same year, he joined Fujitsu Kyushu Communicatin Systems Limited (Fujitsu Kyushu Network Technologies Limited) and engaged in research of communication systems. He is the vice chief director of Kyushu Embedded Software Technology Consortium and executive secretary of ES-Kyushu. He is also the executive secretary of Kyushu branch of the Society of Project Management and program committee of Software Engineers Association.

Shuichi Takayama received his Bachelor's degree and Master's degree from Kyushu Institute of Technology in 1994 and 1996. In the same year, he joined Fujitsu Limited. He is assigned to Fujitsu Kyushu Network Technologies Limited and engages in development of hardware and software for image processing.