ORIGINAL RESEARCH

A study of quality prediction for large-scale open source software projects

Shinji Akatsu*[1], Ayako Masuda[2], Tsuyoshi Shida[2], Kazuhiko Tsuda[1]

[1] *Graduate School of Business Sciences, University of Tsukuba, Tokyo, Japan*
[2] *Research Group on Business Informatics on Social Science, University of Tsukuba, Tokyo, Japan*

Received: April 30, 2020 **Accepted:** February 21, 2021 **Online Published:** April 21, 2021
DOI: 10.5430/air.v10n1p34 **URL:** https://doi.org/10.5430/air.v10n1p34

ABSTRACT

Open source software (OSS) has seen remarkable progress in recent years. Moreover, OSS usage in corporate information systems has been increasing steadily; consequently, the overall impact of OSS on the society is increasing as well. While product quality of enterprise software is assured by the provider, the deliverables of an OSS are developed by the OSS developer community; therefore, their quality is not guaranteed. Thus, the objective of this study is to build an artificial-intelligence-based quality prediction model that corporate businesses could use for decision-making to determine whether a desired OSS should be adopted. We define the quality of an OSS as "the resolution rate of issues processed by OSS developers as well as the promptness and continuity of doing so." We selected 44 large-scale OSS projects from GitHub for our quality analysis. First, we investigated the monthly changes in the status of issue creation and resolution for each project. It was found that there are three different patterns in the increase of issue creation, and three patterns in the relationship between the increase in issue creation and that of resolution. It was confirmed that there are multiple cases of each pattern that affect the final resolution rate. Next, we investigated the correlation between the final resolution rate and that for a relevant number of months after issue creation. We deduced that the correlation coefficient even between the resolution rate in the first month and the final rate exceeded 0.5. Based on these analysis results, we conclude that the issue resolution rate in the first month once an issue is created is applicable as knowledge for knowledge-based AI systems that can be used to assist in decision-making regarding OSS adoption in business projects.

Key Words: Open Source Software, OSS Development, Software Quality, Quality Prediction

1. INTRODUCTION

In recent years, remarkable progress has been made in the development of open source software (OSS). Typical examples of OSS include the web service stack LAMP[1,2]—which, in turn, is composed of four OSS components namely Linux,[3] Apache,[4] MySQL,[5] and PHP[6] / Perl[7] / Python[8]—and the Android[9] operating system. Furthermore, various OSS applications that have been developed for LAMP and Android environments have been made available free of charge along with their source codes.

In contrast, enterprise software (ES) includes software products owned and developed by corporations and their source codes are protected through copyright. Owing to the cost model for ES, its development is focused on development efficiency and number of users. Therefore, improving development efficiency and increasing the number of users are important themes for ES engineering.

Because the use of an OSS is typically free, and its source

*Correspondence: Shinji Akatsu; Email: s1345001@u.tsukuba.ac.jp; Address: Graduate School of Business Sciences, University of Tsukuba, 3-29-1, Otsuka, Bunkyo-ku, Tokyo 112-0012, Japan.

code can be openly modified and redistributed within the scope of its open source license, OSS reuse is common; thus, OSS development is not focused on development efficiency. Therefore, there is no appropriate cost model for OSS development, and research is underway to define and optimize OSS development costs.[10,11]

Nevertheless, the adoption of OSS for software development processes in corporations improves development effectiveness and efficiency; however, important factors need to be considered because OSS licenses do not promote monopolistic use of intellectual property. In particular, the adoption of OSS technologies requires crucial decision-making based on various aspects, including software quality, development investment, business and technology strategies, and intellectual property management, which are not mutually independent, but instead might be related to each other in a complex manner. In our previous study, we presented a structured analysis approach to separate evaluation criteria and their contributing factors for OSS-adpoted software development and attempted to clarify the structured evaluation criterion map.[12]

As specified above, evaluation of software quality is an important factor in the decision-making process for OSS adoption. In general, depreciation principles of accounting define the useful life of a software as 3 or 5 years.[13] During this depreciation period, software support to resolve bugs and improve specifications is required for OSS as well. However, in practice, for OSS, it is expected that the OSS developer community will provide software support services instead of a corporate enterprise. In recent corporate software development projects, most are based on OSS and it is very rare that software projects do not incorporate OSS somewhere in the process.

From a software-engineering point of view, the functionality and quality of the software are the most important factors for selecting an appropriate system. The functionality of OSS can be verified by analyzing the freely available source code. However, as the deliverables of an OSS are developed by the developer community, their quality is not guaranteed. As the quality of a corporate software product must be guaranteed, the corporate user choosing to adopt the OSS is responsible for evaluating and guaranteeing its quality. Hence, a method for predicting the quality of OSS before it is implemented would be greatly valuable to corporate users.

In general, software that is extensively used tends to be of better quality as more people do testing and reporting bugs. However, there are cases where the actual performance results are not disclosed, and the indicators used to evaluate the quality are unknown. Because OSS development is typically intrinsically motivated, its quality is independent of its cost. Therefore, cost is unlikely to be a good metric to evaluating the quality, and other factors need to be identified. Even if these metrics are not universal, a system for selecting OSS that is recognized as having better quality than competing OSS would be highly advantageous. Furthermore, there have been cases in North America where software development companies acquired the vendors who developed a particular OSS in order to effectively incorporate it as in-house modules into their software products. However, Japanese companies tend to use a desired OSS as provided as third-party modules. Awareness of issues regarding OSS selection seems to be different between Japan and North America. There have been Japanese studies regarding the quality of OSS, but there are few studies from elsewhere. Therefore, it is expected that there would be great demand for a prediction model for evaluating the quality of OSS products.

There are many prior studies regarding OSS in general[11–13] and software quality.[14,15] However, regarding the quality of OSS, prior studies are limited to quality processes on the development side,[16] while there has been no discussion of predicting quality from the perspective of the user, except for a recent study by our research group.[17]

In this study, the quality of OSS is defined as the "resolution rate of the issues processed by OSS developers and the promptness and continuity of handling bugs and the other issues." The objective of this study is to develop an artificial intelligence (AI)-based quality prediction model that corporations could use to assist in deciding whether an OSS should be adopted based on its quality. In particular, the code development records for an OSS can be examined to determine OSS quality in terms of issue resolution rate.

The OSS development data used in this study were obtained from GitHub.[18] To perform statistical analysis and improve the accuracy of analysis results, it is necessary to consider a significant number of OSS development data. Therefore, we extracted 44 large-scale projects that were registered on GitHub in 2012 and were still under development until August 11, 2017. For the 44 extracted projects, the Git repositories included around 17,000 MB of deliverables data, and the total number of issues identified for resolution was approximately 620,000.

In this study, we analyzed the quality of OSS based on the following propositions.

- Proposition 1: For each extracted project, aggregate the status transitions for software issues including creation (open) and resolution (close) of issues on a monthly basis, and determine the characteristics peculiar to OSS development.

- Proposition 2: Analyze the relationship between the final resolution rate and factors that affect it.
- Proposition 3: Examine the cause of the identified peculiar characteristics of OSS development.

The corresponding results obtained in this study are summarized as follows.

- Three patterns of increase in issue creation, and three trends in the relationship between the increases in issue creation and resolution were identified. Multiple cases for each pattern were confirmed during the different resolution periods.
- The correlation between the final resolution rate and resolution rate for the relevant period was analyzed. It was found that the correlation coefficient between the resolution rate for the first month and final rate also exceeded 0.5.
- Based on our analyses, it was observed that, in OSS projects, which are voluntary projects, promptness of bugs and issues resolution was prioritized over activity continuity. It was concluded that the resolution rate for issues in the first month after they are identified is applicable as knowledge for knowledge-based AI systems that can be used to assist businesses with decision-making regarding OSS adoption.

2. Definition of OSS quality and its measurement

In this section, we define the quality of OSS and describe a measurement method for it.

2.1 OSS quality

Track record of adopted implementation is one of the criteria considered by corporations before adopting new software. In the case of ES, the company that develops an ES, also performs thorough testing before deployment; in addition, it fixes bugs or improves software specifications after deployment. Their efforts in regard to quality will lead to adoption results. Therefore, it is assumed that ES typically is of good quality. In contrast, the record of adopted implementation for OSS is not obvious. Thus, in this study, the quality of an OSS is defined based on the manner in which software code and specifications are maintained by OSS development community. Furthermore, the OSS code itself can be checked and issues can be raised by users considering its adoption; therefore, promptness and continuity of maintenance are requirements for software adoption. In particular, status transition of an identified issue is an indicator of quality for such software; this is because a high resolution rate and short resolution time could be assumed as indications of good software quality.

2.2 Quality measurement method

OSS development communities perform issue management for OSS using the "Issues" tracker feature in each repository wherein the project deliverables are stored. We made the following measurements by using their issue management data. First, to determine the resolution rate for an OSS project, the number of issues that transitioned from being created to being closed was measured on a monthly basis. Next, to estimate promptness and maintenance continuity, the transition time from identification to resolution of issues was also measured on a monthly basis. In particular, we summarized the transition time it took to resolve created issues every month.

3. Selection and analysis of target projects

In this section, we describe the selection method based on which the 44 OSS projects were considered for analysis. Moreover, an overview of these projects and the results of our data aggregation are presented.

3.1 Extraction of OSS development data

To analyze OSS quality, it is necessary to obtain a large amount of OSS development data. The source for the analysis data may be GitHub or Bitbucket,[19] which are web sharing services that provide a version control system. In this study, we used OSS development project data published on GitHub.

GitHub uses the version management system Git,[20] and provides the web application program interface (API) GitHub API v3,[21] using which users can access repositories that are directories storing project deliverables and development history, among others. In this study, we used the GitHub REST API to select the target projects as well as survey them. For example, the function "Issues"—the issue management feature in GitHub—returns the timestamp for the issue creation date, status of an issue, and comments related to an issue.

3.2 Target project selection

In this study, sample projects were selected using GitHub API v3 as per the following criteria.

- Extract OSS development projects registered in early 2012—when GitHub started—until August 11, 2017 to ensure that the collected data was long-term data.
- Extract projects whose repository size is 15 MB or more to ensure a large-scale OSS is selected. As a result of surveying companies in terms of the OSS repository size ($\geq$10 MB, $\geq$15 MB, or $\geq$20 MB), 15 MB was the most common, and was hence used as a criterion for judging large-scale OSS development in the software development industry. For example,

a premium automobile contains close to 100 million lines of software code,[22] which is about 10 times that of a 15 MB repository. In a previous study,[23] 15 MB was also used.

- Extract projects whose developers duplicate projects in their own development environment, and these developers have 200 or more forks, to ensure a large-scale OSS is selected.
- Extract projects whose star counts evaluated by OSS users are 1000 or more to ensure the quality of OSS project deliverables.

Next, we selected projects whose contributor number and commit number are within the second quartile or more of those numbers to avoid bias because of considerably few project contributors; this is because the number of contributors indicates whether sufficient human resources were available for a project. Then, for accurate statistical analysis, we excluded projects that have missing values and projects wherein all issues were deemed resolved. Finally, 44 projects were selected.

These selected sample projects are listed in Table 1. In particular, Table 1 shows the project data as of December 31, 2018, which was extracted from GitHub on April 23, 2020. Data as of December 31, 2018 was selected in order to eliminate the influence of unresolved issues between December 31, 2018 and March 31, 2020 on our analysis. In Table 1, the "Created Issues" column represents the total number of issues created in the repository, while the "Closed Issues" column indicates the number of created issues closed after resolution. "Resolution Rate" is the ratio of the number of "Closed Issues" to those of "Created Issues." The total number of created issues included in the selected repositories is about 620,000, which is sufficient for statistical analysis.

3.3 Change in the number of created and closed issues

In order to analyze the overall issue resolution rate for the 44 selected projects, the cumulative number of created and closed issues was determined on a monthly basis. Example transition plots for this cumulative analysis are shown in Figure 1.

Useful information regarding OSS quality can be deduced based on the change in the cumulative numbers of created and closed issues in OSS development. First, three patterns can be recognized from the increasing number of created issues. As shown in the example in Figure 1(a), the increase in the number of created issues is almost linear, i.e., there is a continuous increase in the number of issues during the development period. Then, as shown in the example in Figure 1(b), the increase in the created issues is small in the initial stage, large in the middle stage, and small again in the final stage of the extracted OSS development data. In this case, the increasing trend in the number of issues follows the curve for the third power.

Table 1. Number of created issues, closed issues, and resolution rate for the 44 selected projects as on December 31, 2018

No.	Repository Name	Created Issues	Closed Issues	Resolution Rate
1	alluxio	8,226	8,218	0.9990
2	ansible	50,412	47,431	0.9409
3	atom	18,429	18,068	0.9804
4	bokeh	8,528	8,259	0.9685
5	bolt	7,725	7,692	0.9957
6	bosh	2,108	2,014	0.9554
7	canjs	4,688	4,470	0.9535
8	Cataclysm-DDA	27,376	27,127	0.9909
9	collectd	3,031	2,648	0.8736
10	conda	8,070	7,306	0.9053
11	contiki	2,550	2,032	0.7969
12	core	33,974	32,824	0.9662
13	crystal	7,129	6,566	0.9210
14	DefinitelyTyped	31,807	29,494	0.9273
15	django	10,800	10,743	0.9947
16	druid	6,780	6,425	0.9476
17	Firmware	11,116	11,009	0.9904
18	frontend	20,861	20,835	0.9988
19	habitica	10,912	10,745	0.9847
20	hazelcast	14,333	13,845	0.9660
21	homebrew-cask	56,809	56,800	0.9998
22	Kotlin	2,046	2,018	0.9863
23	libgdx	5,489	5,235	0.9537
24	linux	2,787	2,705	0.9706
25	lodash	4,127	4,125	0.9995
26	meteor	10,373	10,341	0.9969
27	mpv	6,357	6,074	0.9555
28	neo4j	12,109	11,985	0.9898
29	nikola	3,192	3,178	0.9956
30	nixpkgs	53,069	50,965	0.9604
31	opencv	13,547	12,274	0.9060
32	openlayers	9,092	9,081	0.9988
33	phpmyadmin	14,811	14,533	0.9812
34	ppsspp	11,585	10,957	0.9458
35	PrestaShop	11,995	11,467	0.9560
36	presto	12,148	11,670	0.9607
37	radare2	12,616	11,726	0.9295
38	ReactiveCocoa	3,629	3,591	0.9895
39	rethinkdb	6,686	5,316	0.7951
40	RIOT	10,686	10,404	0.9736
41	servo	22,577	20,451	0.9058
42	spring-boot	15,583	15,397	0.9881
43	web-platform-tests	14,664	13,701	0.9343
44	yii2	16,680	16,282	0.9761
	Total	621,512	598,027	

Moreover, as shown in the example in Figure 1(c), the increase in the number of issues was large at the beginning, but decreased at the end, i.e., the number of issues increased

logarithmically. It is noteworthy that the 44 projects selected in our study were fairly successful long-term development projects, and therefore, the number of issues decreases in the later stages of all examples shown in Figure 1. However, in other projects wherein the number of issues increases steadily or as a power of two, their project quality is difficult to determine, and these will be considered in a future work.

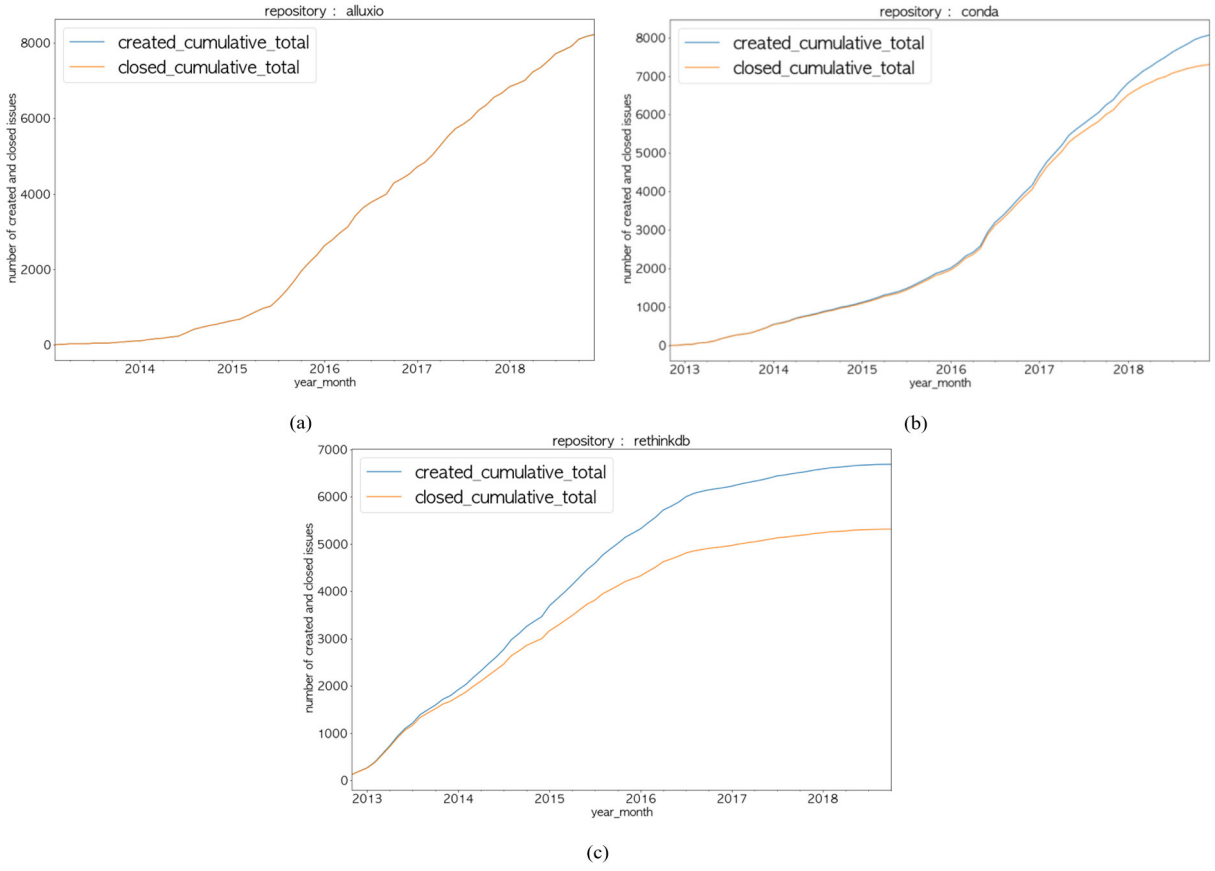

Figure 1. Examples of trends for changes in cumulative number of created and closed issues. (a) Numbers of created and closed issues continuously match each other. (b) Numbers of created and closed issues diverge during the middle stage of development. (c) Numbers of created and closed issues are divergent from the beginning.

Second, the corresponding relationships between the increase in the number of created issues and their resolution are also depicted in Figures 1(a)–(c); these relationships are as follows:

Figure 1(a): Increases in the number of created and closed issues are aligned together.

Figure 1(b): Increases in the number of created and closed issues begin to diverge during the middle stage.

Figure 1(c): Increases in the number of created and closed issues remained separate from the beginning.

In Figure 1(a), a pattern is observed wherein new issues are continuously generated and also continuously solved; consequently, the final resolution rate is high. Figure 1(b) depicts a pattern wherein the initial resolution rate is in line with the increase in created issues, but the issue resolution frequency slows down during the middle development phase, and thus, the final resolution rate gradually decreases. Finally, Figure 1(c) shows a pattern wherein the number of created issues exceeded the number of resolved issues from the beginning; therefore, the final resolution rate is relatively low.

Because the three types of patterns described above are derived based on the cumulative numbers of created and closed issues, it is difficult to determine if there are many unresolved issues or time-consuming ones in a specific project, which then lead to a decrease in the overall resolution rate. Thus, in the next section, we investigate the development stage wherein such issues were created and study the distribution of the time required for their resolution.

4. QUALITY PREDICTION MODEL BASED ON ISSUE RESOLUTION RATE

Here, we analyze the timestamp information of created issues and the duration required to resolve these issues. Then, we examine this data to derive knowledge that can be applied to the proposed AI-based quality prediction model.

4.1 Trends in monthly resolution status

To understand resolution promptness and maintenance continuity, we measured the time in which a created issue was closed every month based on the data of the 44 selected projects; in particular, we investigated the transition period for issue resolution status on a monthly basis. Example trend plots for this resolution status analysis are shown in Figure 2. The following observations were made for the three examples discussed in Section 3.3.

The pattern in Figures 2(a-1) and (a-2) indicates a high resolution rate. In the case shown in Figure 2(a-1), the created issues are typically resolved in the same month they are created, while, in the case shown in Figure 2(a-2), the issue resolution period is longer, but an issue is typically closed within 12 months. The pattern in Figure 2(b) indicates that the resolution rate slows down during the middle period of OSS development; this is because there are cases wherein unsolved issues gradually accumulate during the middle phase leading to longer resolution period for created issues. Finally, Figures 2(c-1) and (c-2) show patterns wherein the resolution rate was slow from the beginning of OSS development. In particular, in the case shown in Figure 2(c-1), issues seemed to take a relatively long time to resolve in the beginning, and thus, unresolved issues overflowed into the middle development period.

4.2 Resolution analysis summary

With regard to the creation and resolution of issues in each project, we investigated the transition in the statuses of issues on a monthly basis and confirmed that OSS development had the following characteristics:

- In terms of issue resolution rate, it was confirmed that there are three types of resolution patterns that affect the final resolution rate.
- It was also confirmed that different cases of resolution promptness and maintenance continuity lead to different final resolution rates.
- It was assessed that a final resolution rate can be predicted based on the resolution rate in the later stages of OSS development. However, it is possible to predict the final resolution rate based on the status of issue resolution in the early stages of development.

4.3 Derivation of knowledge for prediction of final resolution rate

As discussed previously, the quality of an OSS is an important issue when its adoption is being considered for business projects. Though there are various quality indicators, prompt and continuous issue resolution are two indispensable ones. Therefore, it is necessary to observe two operations, namely occurrence of and response to events.

Occurrences and responses that can be observed in the current OSS include creation and resolution of issues. While issues are not necessarily limited to bugs or quality, most issues are related to bugs after all. In addition, software issues affect terms of use and consultation items regarding usage, which, in turn, affect sales activities of business products as well as the role of customer care. Though these factors also determine the usefulness of an OSS in business projects, analyzing the resolution time for issues and number of issues still help understand the quality of an OSS.

Therefore, to determine whether an OSS would be appropriate for a business project, we decided to observe the resolution time and number of created and resolved issues. We observed that it is not so difficult to judge the appropriateness of an OSS after analyzing the transitions of created and resolved issues in each considered OSS project over many years. For example, on observing the cumulative transitions of created and closed issues of three projects as depicted in Figure 1, we can determine the comparative quality of the different OSS. Furthermore, human evaluator bias will not distort these results.

However, not all OSS projects necessarily accumulated development years enough for prediction of quality by long-term observation. There are quite a few OSS which business corporate desire to adopt regardless of short development history. Therefore, we investigated whether it is possible to roughly grasp the resolution time and number of created and resolved issues in a few years in the future by observing the number of created and resolved issues for a certain number of months after the deliverables of the project are made available.

Thus, in order to predict the final issue resolution times and number of issues for the 44 selected projects, the distribution of the number of months it took for each issue in the project from creation to resolution was calculated. Then, the correlation between the above-mentioned distribution and the approximate final resolution status was examined. Table 2 lists our calculation results for the correlation between the resolution rate of issues at the nth month after each issue is identified and the final resolution rate of all 44 projects.

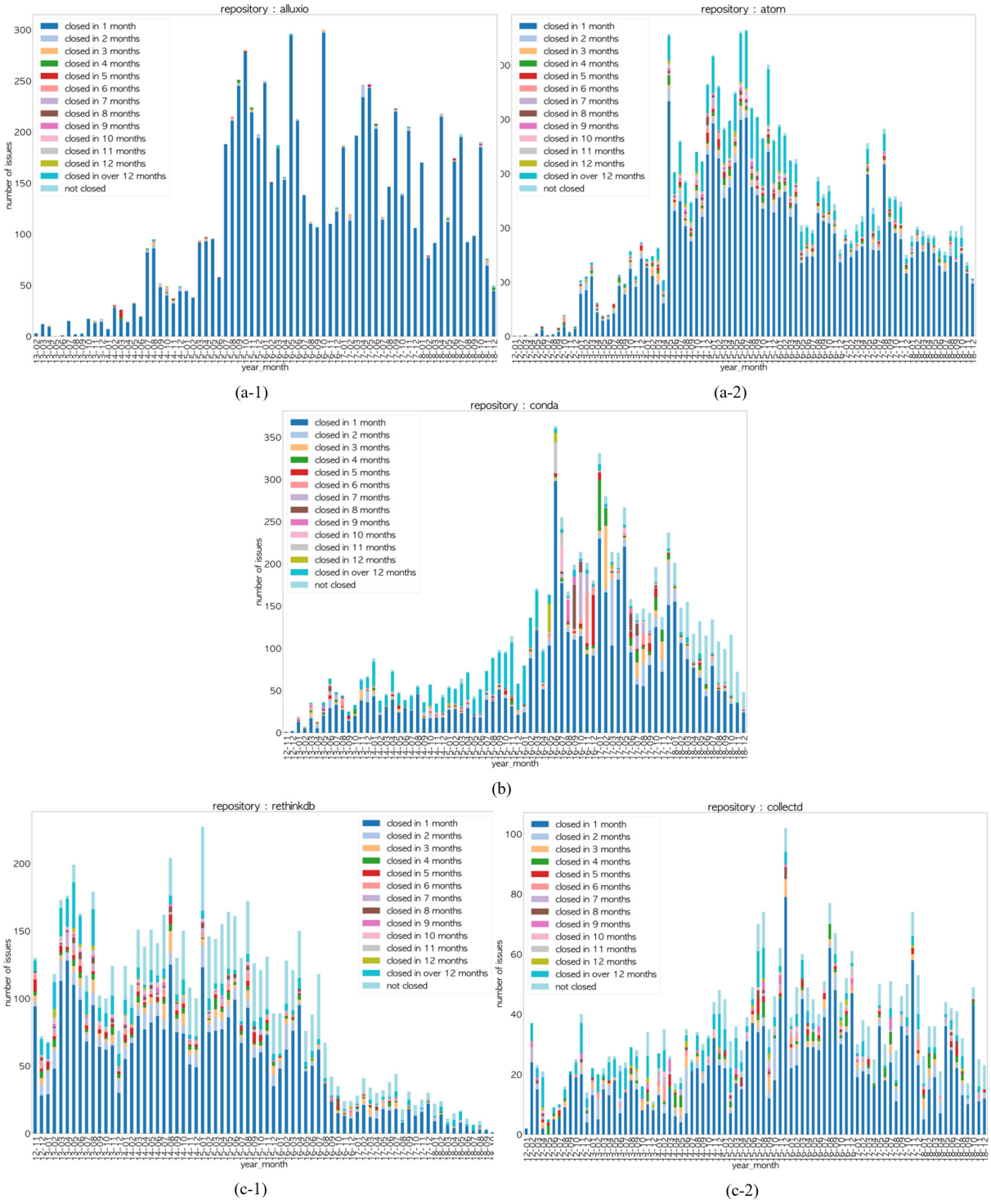

Figure 2. Patterns of transition in resolution status for each month by period. (a-1) Created issues continue to be resolved in the current month. (a-2) Resolution period for the issues is extended; however, they are consistently closed within 12 months. (b) Unresolved issues are present as new issues are created, thus extending from the middle stage to end of the relevant period. (c-1) Issue took considerable time to resolve. Unresolved issues from the middle stage are brought forward. (c-2) Issues continuously created and take time to resolve leading to accumulation of unresolved issues.

Table 2. Correlation between final resolution rate and resolution rate for a relevant number of months after issue creation

Months for Resolution	Correlation Coefficient
1	0.502241981
2	0.485930706
3	0.474072147
4	0.465699399
5	0.458041663
6	0.451808318
7	0.450845338
8	0.450177635
9	0.450427825
10	0.449877651
11	0.44847918
12	0.447750932

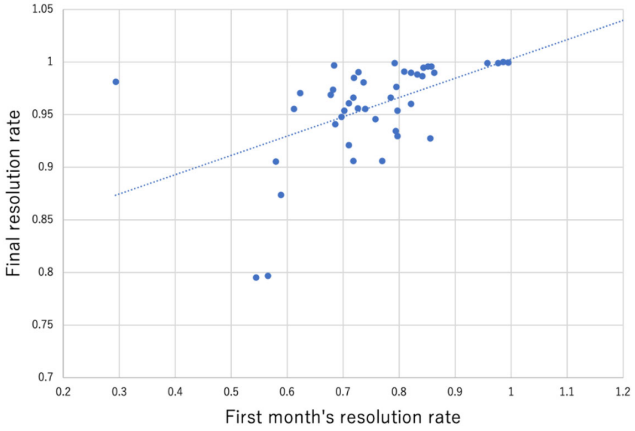

Figure 3. Correlation between final resolution rate and that of the first month after issue creation

It is clear from the results in Table 2 that, at the end of the first month, the correlation coefficient between the final resolution rate and that of the month exceeded 0.5, which is an extremely high value. Furthermore, this correlation in the final resolution rate does not change even when the observation period is increased. Figure 3 shows the correlation between the resolution rates in the first and final observation month for all 44 projects. From the figure, it can be observed that a project that responds well to each issue at an early stage also responds well to them at the end, i.e., it can be said that it is sufficient to analyze the correlation between open and resolved issues in the first month in order to decide whether to adopt an OSS in a business project.

5. Conclusions

In this study, we selected 44 large-scale OSS projects from GitHub for our analysis to deduce the quality of an OSS and determine if it will be suitable for adoption in a business project. First, we investigated the monthly changes in the status of issue creation and resolution for each project. It was found that there are three patterns in the increase in issue creation as well as three patterns in the relationship between the increase in issue creation and that of resolution. Based on our analysis, we confirmed that there are multiple cases of each pattern that affect the final resolution rate.

Next, we investigated the correlation between the final resolution rate and resolution rate for a relevant number of months after issue creation. We observed that the correlation coefficient even between the resolution rate in the first month and the final rate exceeded 0.5. Therefore, it can be concluded that the issue resolution rate for the first month is suitable as knowledge for knowledge-based AI systems, which, in turn, can be used to assist in decision-making regarding OSS adoption in business projects.

Because information technology is being constantly improved, an increasing number of useful OSS are being developed as well. Therefore, in the near future, the adoption decision for latest OSS will have to be made with a short track record. Thus, a possible future work will involve the derivation of knowledge based on which the final quality of an OSS can be predicted from the initial response status after the project is launched.

References

[1] Ware B. Open source web development with LAMP: Using Linux, Apache, MySQL, Perl, and PHP. Addison-Wesley Longman Publishing Co., Inc.; 2002.

[2] Gerner J, Naramore E, Owens M, et al. Professional Lamp: Linux, Apache, MySQL and PHP5 Web Development. John Wiley & Sons; 2005.

[3] Linux [Internet]. [cited 2020 Mar 26]. Available from: https://www.linux.com/

[4] Apache [Internet]. [cited 2020 Mar 26]. Available from: https://www.apache.org/

[5] MySQL [Internet]. [cited 2020 Mar 26]. Available from: https://www.mysql.com/

[6] PHP [Internet]. [cited 2020 Mar 26]. Available from: http://php.net/

[7] Perl [Internet]. [cited 2020 Mar 26]. Available from: https://www.perl.org/

[8] Python [Internet]. [cited 2020 Mar 26]. Available from: https://www.python.org/

[9] Android [Internet] [cited 2020 Mar 26]. Available from: https://www.android.com/

[10] GitHub: The largest open source community in the world [Internet]. [cited 2020 Mar 26]. Available from: https://github.com/open-source/

[11] Kuwata Y, Ishizuka T, Yokoyama S, et al. A study on a cost model of OSS community and the optimization of operation of operation cost. 20th Study Group of Knowledge Sharing Network. SIG-KSN. Vol. 20, No. 7. The Japanese Society for Artificial Intelligence (2017).

[12] Akatsu S, Fujita Y, Kato T, et al. Structured analysis of the evaluation process for adopting open-source software. Procedia Comput. Sci. 2018 Jan; 26: 1578-86. https://doi.org/10.1016/j.procs.2018.08.131

[13] No. 5461 Acquisition cost and useful life of software [Internet]. National Tax Agency [cited 2020 Apr 29]. Available from: https://www.nta.go.jp/taxes/shiraberu/taxanswer/hojin/5461.htm/

[14] Tosun A, Bener A, Kale R. AI-Based Software Defect Predictors: Applications and Benefits in a Case Study. Proceedings of the Twenty-Second Innovative Applications of Artificial Intelligence Conference (IAAI-10). 2010.

[15] Radlinski L. A conceptual Bayesian net model for integrated software quality prediction. Annales UMCS Informatica AI XI. 2011; 4: 49–60. https://doi.org/10.2478/v10065-011-0032-5

[16] Bahamdain SS. Open Source Software (OSS) Quality Assurance: A Survey Paper, Procedia Computer Science, 2015 – Elsevier. https://doi.org/10.1016/j.procs.2015.07.236

[17] Akatsu S, Masuda A, Shida T, et al. A Study of Quality Indicator Model of Large-Scale Open Source Software Projects for Adoption Decision-Making. Procedia Comput. Sci. 2020; 176: 3665-72. https://doi.org/10.1016/j.procs.2020.09.020

[18] GitHub [Internet]. [cited 2020 Mar 26]. Available from: https://github.com/

[19] Bitbucket [Internet]. [cited 2020 Mar 26]. Available from: https://bitbucket.org/

[20] Git [Internet]. [cited 2020 Mar 26]. Available from: https://git-scm.com/

[21] GitHub API v3 [Internet]. [cited 2020 Mar 26] Available from: https://developer.github.com/v3/

[22] Charette RN. This car runs on code. IEEE Spectrum, 2009 Feb.

[23] Masuda A, Matsuodani T, Tsuda K. Team Activities Measurement Method for Open Source Software Development Using the Gini Coefficient. 2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW). 2019: 140-147.