Study on Adopting Open Source Software in Business Projects

Study on Adopting Open Source Software in Business Projects

２０２１

Shinji Akatsu

１１〇１１１

Shinji Akatsu

Study on Adopting Open Source Software in Business Projects

２０２１

Shinji Akatsu

Graduate School of Business Sciences
University of Tsukuba

Abstract

Recently, adopting open-source software (OSS) in the software development process, has become wide-spread among enterprise system development projects. It contributes to improving development effectiveness; however, it requires important factors to be considered, such as it does relate to business alliance strategy including the platform strategy in global ecosystems, it does not insist on the monopolistic use of intellectual property and its does not guarantee its software quality, and so on. This implies that adopting OSS requires crucial decision making in terms of various aspects including business, technology, and intellectual property management, which are not mutually independent but may exhibit a complex set of relationships.

Therefore, considering the situation in which the use of OSS is indispensable in corporate system development, the subject of this dissertation is to identify issues for adoption evaluation and to propose technologies to support decision-making regarding adopting OSS in business software projects. The objective is to contribute to the efficiency of the system development of enterprises.

First, we articulate the first research question as “Isn't it possible to clarify the decision-making procedure by extracting the axes and factors of OSS adoption evaluation and by creating a structured map to overview the OSS to be adopted?”

The significance of this study is its organization of the concept of the case-by-case process in decision-making for OSS adoption of business projects in the actual workplace.

Thus, a method for organizing the evaluation axes and factors of OSS adoption evaluation and the structure of the evaluator hierarchy is assessed. A structured map is proposed that defines the positioning of the OSS evaluated from the perspective of technology, products, business project, company operation, and intellectual property creation and application in order for project managers responsible for the entire system development to obtain an overview of the adoption evaluation.

In the practical consequences of the company, the adoption is decided based on the importance priority of the desired OSS, including, for example, the function, the quality and/or the intellectual property, however if the positioning is the same, the OSS with better quality will be adopted. Therefore, we decided to dig deeper into what and how OSS software quality, which is a key factor in hiring decisions, should be evaluated. Here, from the viewpoint of OSS usage, we define the quality of OSS as "the resolution rate of issues processed by OSS developers as well as the promptness and continuity of doing so."

Second, we articulate the second research question as "By looking at the status of the issue session of the OSS development projects by the OSS developer community, is it possible to extract an index that can quickly determine whether OSS can be used before starting a detailed examination?"

The significance of this research lies in the proposal of a framework for adoption decision making from an OSS quality perspective. Software quality indicators, which are a key factor in adopting OSS, should be explored in depth.

Thus, from the analysis of issue sessions of the OSS development projects, a nine-quadrant map named T-model is defined, focusing on the trends expressed by the curve shape and the divergence time, in terms of the

cumulative number of issues raised and resolved. Then the mapped OSSs in the T-model are examined from the perspective of the final resolution rate. The axes to be related to software quality are the activeness of the developer community and its maturity of the technological innovation. It is proposed to utilize the T-model as an index to quickly judge whether the target OSS can be used before starting a detailed examination.

Third, we articulate the third research question as “By looking at the resolution rate and the response quickness and the response continuity of the issue session of the OSS development projects by the OSS developer community, is it possible to extract the knowledge to predict the final quality including the support capability in actual use?”

The significance of this research is to quantitatively analyze the actual activities of the development community and to extract the knowledge, from the perspective of digging deeper into the axis of the activeness of the OSS development community. The indicator to predict the OSS software quality including the support capability in actual use, which is a key factor in adopting OSS, should be explored in depth.

As a result, the resolution rate of the OSS project’s issue response and the status of promptness and continuity are analyzed, as well as the resolution rate of the issues at the early stage (first month) regarding quality including support capability in actual use. There is a high correlation with the final solution rate, which meant that it could be used as knowledge for introduction decisions.

In this dissertation, a method to define the positioning of the target OSS in the OSS adoption evaluation is proposed. The model consists of two axes, which are the activeness of the community and its technological maturity, is proposed to quickly judge based on software quality whether the target OSS

can be used before starting a detailed examination. The knowledge to judge better quality OSS, considering support capability in actual use from the perspective of the OSS usage side, is extracted. By utilizing this method and model and knowledge in the decision-making process for adopting OSS, we believe that it will contribute to improving the productivity and efficiency of system development in business projects.

Contents

List of Tables

List of Figures

1. Introduction

Recently, adopting open-source software (OSS) in the software development process, has become wide-spread among enterprise system development projects. It contributes to improving development effectiveness; however, it requires important factors to be considered, such as it does relate to business alliance strategy including the platform strategy in global ecosystems, it does not insist on the monopolistic use of intellectual property and its does not guarantee its software quality, and so on. This implies that adopting OSS requires crucial decision making in terms of various aspects including business, technology, and intellectual property management, which are not mutually independent but may exhibit a complex set of relationships.

Therefore, considering the situation in which the use of OSS is indispensable in corporate system development, the subject of this dissertation is to identify issues for adoption evaluation and to propose technologies to support decision-making regarding adopting OSS in business software projects. We believe that the required items include an organized concept of the vague behavior of the actual workplace, and a framework for adoption decision making from an OSS quality perspective based on the quantitative analysis of the actual activities of the OSS development community to extract the related knowledge. The objective is to contribute to the efficiency of the system development of enterprises.

Based on the issues of related research, the following three questions are articulated as the research themes.

(1) Research question 1: “Isn't it possible to clarify the decision-making

procedure by extracting the axes and factors of OSS adoption evaluation and by creating a structured map to overview the OSS to be adopted?"

(2) Research question 2: "By looking at the status of the issue session of the OSS development projects by the OSS developer community, is it possible to extract an index that can quickly determine whether OSS can be used before starting a detailed examination?"

(3) Research question 3: "By looking at the resolution rate and the response promptness and the response continuity of the issue session of the OSS projects by the OSS developer community, is it possible to extract the knowledge to predict the final quality including the support capability in actual use?"

Regarding the first research question, it is necessary to have a method to organize the evaluation axis and factors of OSS adoption evaluation and the structure of the evaluator hierarchy and to define the positioning of the OSS to be evaluated from various aspects, such as technology, including quality and function; business, including alliance strategy; and intellectual property management, including its creation and utilization.

In the practical consequences of the company, the adoption is decided based on the importance priority of the function and/or intellectual property of the desired OSS; however, if the positioning is the same, the OSS with better quality will be adopted. Therefore, it is necessary to identify what and how OSS software quality, which is a key factor in hiring decisions, should be evaluated. Considering the viewpoint of OSS usage, we define the quality of OSS as "the resolution rate of issues processed by OSS developers as well as the promptness and continuity of doing so."

Regarding the second research question, what is required to determine quickly whether OSS can be used before starting a detailed examination, is a framework for adoption decision making from an OSS quality perspective–with OSS quality indicators being of major significance – which is an important factor of OSS adoption evaluation. This should be extracted from the analysis of the activity of OSS development projects by the OSS developer community, as the activeness of the developer community and its maturity of the technological innovation seems to be related to software quality.

Regarding the third research question, what is required is the knowledge of predicting software quality including the support capability in actual use, based on quantitative analysis of the actual activities of the development community. It is preferable to predict the final quality–in particular, the final resolution rate–based on the activity status in the early past stage of OSS development projects by OSS developer continuity.

The structure of this dissertation is as follows. In Chapter 2, the position of our study is clarified via an overview of OSS adoption in business projects and by surveying related previous studies from the viewpoint of the evaluation process for adopting OSS, OSS development project challenges, and software quality modeling and prediction of OSS. In Chapter 3, a concept of the decision-making process for adopting OSS is organized by identifying the axes and factors of OSS evaluation and by defining a structured map. In Chapter 4, a framework for adoption decision making from an OSS quality perspective is proposed. The proposed model shows that the activeness of the developer community and its maturity of the technological innovation are important indicators in terms of OSS quality. In Chapter 5, to dive deep into the activeness of the community, knowledge for predicting the software

quality, including the support capability of the developer's community, is extracted. Finally, in Chapter 6, the conclusion is presented.

2. Issues on Open Source Software Adoption in Business Projects

In this chapter, we explain the overview of OSS adoption in business projects and the survey results of prior related studies and also outline the key contributions that are aimed for.

2.1. Overview of Open Source Software Adoption in Business Projects

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 [Ware 2002] [Gerner et al. 2005]—which, in turn, is composed of four OSS components namely Linux [Linux 2020], Apache [Apache 2020], MySQL [MySQL 2020], and PHP [PHP 2020] / Perl [Perl 2020] / Python [Python 2020]—and the Android [Android 2020] 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 the 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 code can be openly modified and redistributed within the scope of its open source license [OSI 2013], 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 [Yoshitaka et al. 2017].

The concept of open source software (OSS) was defined by the Open Source Initiative (OSI) [OSI 2013]. The source code is open to the public, and regarding use and distribution, the OSS license requires various freedoms (free redistribution, etc.) and obligations and restrictions (distribution in source code, allowing modifications and derived works, etc.) for each OSS [OSI 2013]. The OSS developer community is responsible for development, testing, maintenance, and so on [OSI 2013]. The motivation for this effort has traditionally been volunteer based [OSI 2013], but in recent years, there has also been strategic OSS implementation launched by IT enterprises [Android 2020] [Tatsumoto 2021].

Here, we briefly review value creation in software development, as shown in Figure 2-1. For closed-source software development, which is performed within a company, a company's technology is the input, and the output is the software to be used for business and accumulated as company assets. When OSS is adopted in software development, the OSS, which is a third-party software, is also included as the input to improve efficiency at the points of the quality, development cost, and period. As a side effect, there are limitations depending on the license of the OSS, such as disclosure of the source code, which affect the output.

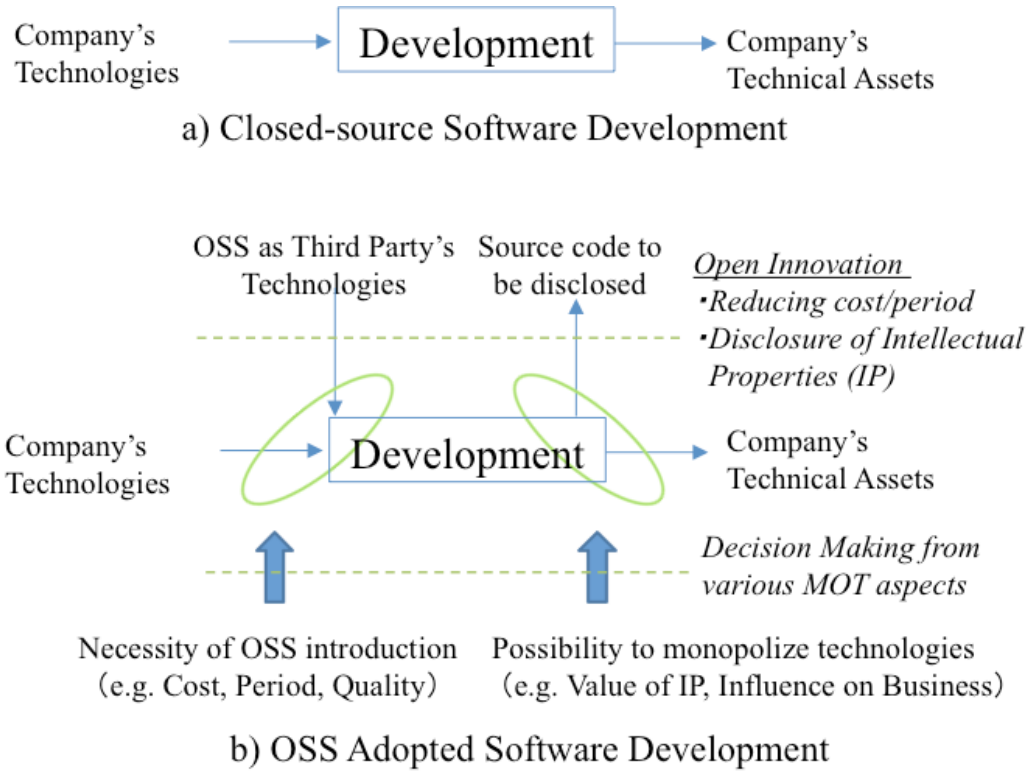

Figure 2-1. Adopting OSS in the Software Development Process

Nevertheless, the adoption of OSS for software development processes in corporations improves development effectiveness and efficiency; however, it requires important factors to be considered, such as it does relate to business alliance strategy including the platform strategy in global ecosystems, it does not insist on the monopolistic use of intellectual property, and its does not guarantee its software quality, and so on [Tatsumoto 2021].

As above mentioned, the adoption of OSS technologies requires crucial decision-making based on various aspects, including business and technology strategies, development investment, software function, software quality, and intellectual property management, which are not mutually independent but instead might be related to each other in a complex manner.

In the practical consequences of the company, the adoption is decided based on the importance priority of the function and/or intellectual property of the desired OSS. However, if the positioning is the same, the OSS with better quality will be adopted because the software quality of the system, including the OSS itself, must be guaranteed by the company who adopted the OSS. OSS quality, including the technical support ability in a broad sense, is completed by its OSS development projects. OSSs are generally developed and operated by a software development team consisting of multiple voluntary engineers. To proceed with software development efficiently with better quality, it is important to maximize the ability of the team.

The issues regarding OSS adoption in business projects at the ground level are summarized as follows:

- The decision to adopt OSS is important in the upstream process of development, but the procedure is case by case depending on the position and priority of the decision-maker.
- The OSS with better quality is selected among the OSSs with the same functionality and intellectual property license, but its quality must be guaranteed by the user.

Therefore, we recognize that organizing the concept of the decision for adopting OSS and evaluation indicators using quantitative analysis is required, also these are an important research theme.

In the following sections, we describe the survey result of the evaluation process for adopting OSS, the fact-finding analysis of OSS development projects, and OSS software quality from the OSS user perspective. The reasons for choosing these items are to explore the research points regarding, OSS adoption judgment, OSS development communities that create the functions and quality of OSS, and OSS quality from the user's perspective.

2.2. Trends and Previous Studies on Evaluation Process for Adopting OSS

Since the adoption of OSS is an action that deals with engineering objects, its evaluation process is a domain where Management of Technology (MOT) is applied.

Here are the general evaluation criteria of software adoption for an enterprise software system based on the roles and viewpoints of the MOT [Nobeoka 2006] and the axis of the integrated roadmap of MOT [Tofu 2011]. In terms of the roles and perspectives of MOT, the major indexes to be considered include technology, customer needs, competitive environment, differentiation and identity, enterprise business structure, and organizational structure [Nobeoka 2006]. Technologies, products, and businesses are adopted as the axis of the integrated roadmap [Tofu 2011]. From these points of view, MOT perspectives include as the key evaluation criteria axis.

- Technology: including function differentiation/uniqueness and quality
- Products: including customer needs
- Business Projects: including competitive environment, differentiation, and distinctiveness
- Company Operation: including organization, investment, policy

Moreover, from an OSS-specific perspective, intellectual property management must be considered because OSS is third-party property as a deliverable by the developer community. Regarding intellectual property management, the cycle of intellectual creation [CabinetSecretariat 2007], consisting of the creation, acquisition, and utilization of intellectual property, is important. In the case of adopting OSS, the trade-off between the creation of one's own property and the utilization of third-party property must be

considered. It is necessary to examine the advantages and disadvantages of the utilization of third-party intellectual property, such as the possibility of enclosing the technology of the deliverable and restrictions on owned or created patents, by carefully examining the license of the OSS. For example, licensing agreements approved by the open-source initiative (OSI) [OSI 2013], such as BSD, GPL, and Apache licenses [BSD-2 2021] [BSD-3 2021] [GPL 2021] [Apache-License 2021], provide a general definition of open source; however, different requirements are stipulated on the terms of patent rights for the disclosure of the source code of derivative works (such as license suspension due to patent litigation) [JapanPatentAttorneysAssocitation 2006]. Therefore, it is necessary to deal with the software depending on the OSS adopted.

Another important OSS-specific perspective is business alliance strategy. The motivation of the OSS developer community's effort has traditionally been volunteer based, but in recent years, there has also been strategic OSS implementation by IT enterprises. Some developer groups are launched by enterprises. When OSS is introduced as a platform, applications and business models may be decided based on choices [Tatsumoto 2021]; therefore, it is also necessary to consider monetization by forming alliances considering network externalities. When combining one's own technology with an open technology, we clarify whether the purpose is to supplement or reinforce one's technology and discuss its necessity. As the interface of joining is required to design with open and closed clarity, consideration of feasibility is necessary.

One topic related to alliance strategy to be considered is the industry direction, which is value creation through "product" and "service" integration. Regarding the business model that connects a product to the Internet and

adds value via web services, it is key to efficiently construct the communication section, which is a commonly used technology, and to focus on developing the core services of the distinctive business itself. There are many OSSs for communication and OSS coupling is important for the product (mono) emphasis on service (koto) [Vargo and Lusch 2008] [Higashi2009] [Chesbrough 2010]. When services became more valuable, connecting functions became important, and OSS was used in large numbers, so the decision to introduce OSS became more important.

Regarding decision-making in general software development, there are common norms such as “Common Frame 2013” [IPA 2013] and “A Guide to the Business Analysis Body of Knowledge” [IIBA2015]. The role of the development process, the content of the role, and the task are specified.

As mentioned above, although there are studies related to the adoption of OSS, these differ from the viewpoint of clarifying the axis of the idea of OSS adoption decision and organizing the concept.

2.3. Trends and Previous Studies on fact-finding of OSS Development Projects

OSS is generally developed, tested, and maintained by the OSS developer community [OSI 2013]. The motivation for this effort has traditionally been volunteer based [OSI 2013]. In recent years, there has been a strategic OSS implementation by IT companies [Android 2020] [Tatsumoto 2021], but the motivations of members of the developer community are just as important. In order to proceed with software development efficiently, it is important to maximize the ability of the team. Previous studies on this theme include the following.

The theme of revitalizing the software development team is analyzed in Masuda's doctoral dissertation [Masuda 2019] from three perspectives: project managers' cognition and team characteristics, the measurement for development efficiency, and the team activities measurement method.

First, regarding project managers' cognition and team characteristics, there is a study that showed high discriminant analysis results regarding the success or failure of a project [Matsuodani 2014]. This is based on a questionnaire survey administered to leaders of corporate development teams. In Masuda's research, the leader of a company's development team proposes a tool verification and optimal analysis method that can know the state of their own team [Masuda 2017].

Second, regarding the measurement of development efficiency, there is a Constructive Cost Model (COCOMO) as a cost model [Boehm 1981]. COCOMO has a proven track record with no restrictions on language or target area. Masuda's research aims to explore the actual situation of the development team based on the relationship between the members and the amount of development in OSS development [Masuda et al. 2018]. Assuming that there is a relationship between the members and the amount of

development of OSS development, an analysis based on a cost model using the power formula in ES development and an attempt to analyze the characteristics of OSS development were conducted. COCOMO was used for the analysis.

Third, for the team activities measurement method, there is a study in which the state of the development team is indexed from the variation in the amount of activity [Masuda 2019].

OSS development is basically volunteer based. Therefore, it is stated that acquiring excellent contributors is a factor in project success because each development process, including software requirement specification development, external and internal specification determination, code writing, testing, and user support, are all volunteer oriented.

The theme of acquiring contributors is analyzed in Kobayakawa's doctoral dissertation [Kobayakawa 2020] from three perspectives: influencers, future potential, and contribution guidelines.

First, regarding influencers, there is research on indicators of influence. The study by Blincoe used the number of followers as an indicator of influencer influence [Blincoe et al.2016]. The study by Badashian argues that the number of "Forks" and "Watches" in a project (owned by the user) should be considered along with the number of followers [Badashian 2016]. The study by Thung et al. used the PageRank algorithm and selected the score as an indicator of influencer influence [Thung 2013]. Some research has also been conducted on whether a lot of contribution activities are needed to become an influencer and other research on whether influencers get more contributors. In Kobayakawa's study, the three patterns above were applied to a cryptocurrency project, and the influence of influencers of cryptocurrency domains was analyzed [Kobayakawa and Yoshida 2019].

Second, regarding future potential, there is a survey study that states that the future of the project leads to the acquisition of contributors [Dabbish et al.2012]. In Kobayakawa's study, the causal relationship with the number of contributors was analyzed using the market capitalization of virtual currency as a proxy variable for the future by time series analysis [Kobayakawa et al. 2020].

Third, regarding contribution guidelines, GitHub recommends that each project have a standard file called the Contribution Guidelines. There is a survey showing the top 50 projects with the highest contributions (committees) that reported that 46 of them had guidelines or websites describing their contributions [Izquierdo et al. 2015]. In Kobayakawa's study, the description of the contribution guidelines was analyzed deeply, and then the analysis method was improved and extended using the structural topic model [Kobayakawa and Yoshida 2017].

We believe that sufficient prior research has been done in this area.

2.4. Trends and Previous Studies on OSS Software Quality

As specified in section 2.2, 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 [NationalTaxAgency 2020]. 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.

Thus, before examining the detailed deliverables of an OSS thoroughly, it is requited to have the knowledge, i.e., indicator for decision, that corporate business could use for quantitatively assessing whether a desired OSS should be adopted.

Moreover, for the further evaluation after the first examination, a method for predicting the quality of OSS before it is implemented would be greatly valuable to corporate users.

In general, widely used software tends to be of better quality as more people

do testing and report bugs. However, there are cases in which 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 in 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.

The prior studies surveyed from this perspective include the following subjects: OSS in general, software quality in general, OSS quality, and OSS usage.

There are many prior studies regarding OSS in general. [Yoshitaka et al. 2017] define the cost model of OSS development as the overall cost from the OSS development community via the service provider to the user and showed the approach to optimize it as OSS development costs. [JapanPatentAttorneysAssocitation 2006] did survey research of intellectual property and licensing agreements perspectives. It showed that different requirements are stipulated on the terms of patent rights for the disclosure

of source code of derivative works (such as license suspension due to patent litigation). Therefore, it is necessary to deal with the software depending on the OSS adopted. [NationalTaxAgency 2020] studied depreciation principles of accounting to define the useful life of a software as 3 or 5 years. During this depreciation period, software support to resolve bugs and improve specifications is required for OSS as well. [Krogh et al. 2012] and [Izquierdo et al. 2015] analyzed the motivation to contribute to OSS development projects.

There are many prior studies regarding software quality by using AI technology. [Tosun et al. 2010] presented AI-based software defect predictors and its applications and benefits in a case study of telecommunication software. [Radlinski 2011] showed a software quality model consisting of multiple conventional quality features such as functional suitability, performance efficiency, operability, security, and compatibility. It is based on a conceptual Bayesian network of an expert type. [Xing et al. 2005] proposed a method for early software quality prediction based on a support vector machine and examined using software for a medical imaging system.

There are several prior studies regarding the quality of OSS. [Bahamdain 2015] did survey research on quality assurance processes on the OSS development side.

Regarding the usage of OSS, there is research on risk management in using it [Franch et al. 2013]. In addition, there are documents published as usage guidelines regarding OSS maintainability and contracts [IPA 2005] [LinuxFoundation 2011].

As mentioned above, there are studies on general software quality and software quality from the perspective of OSS development. However, they are studied from different objectives than ours, which are to define and analyze

OSS quality from the perspective of the OSS user side.

2.5. Chapter Conclusion

This chapter firstly summarizes the overview of OSS adoption in business projects. Then, the trends and past studies on the key elements of OSS adoption in business projects are surveyed. These include the evaluation process for adopting OSS, OSS development projects (revitalizing a team and acquiring contributors in a developer community), and OSS software quality modeling and prediction. Although there are studies related to the adoption of OSS, these differ from the viewpoint of clarifying the axis of the idea of OSS adoption decision and organizing the entire concept. Regarding fact-finding studies on OSS development projects, we believe that sufficient prior research has been done. There are studies on general software quality and software quality from the perspective of OSS development. However, they have been studied from different objectives than ours, which are to define and analyze OSS quality from the perspective of the OSS user side.

Since the use of OSS is indispensable for corporate system development, the objective of this research is to contribute to the efficiency of corporate system development by extracting the challenges related to the OSS adoption decisions and by proposing the methods to support decision-making.

The first challenge in an enterprise software project willing to adopt OSS is that there are a wide variety of factors to consider, making it difficult to make decisions. It is required that the studies have structured the procedure for considering OSS adoption. As shown in section 2.2, there have been no previous studies about this theme to the extent we have investigated. There are similar studies, but no matching ones. Thus, we felt conducting this study would be valuable. Based on the axis and factors of OSS introduction evaluation, it is useful to examine the order of consideration of the factors in

the axis and to clarify the decision-making procedure by the structured map. This theme is studied in Chapter 3.

The second challenge in an enterprise software project willing to adopt OSS is that how to index software quality. Since the user who decides to adopt OSS is responsible for the quality of OSS, the index that can quantitatively judge whether or not it is available OSS is required. As shown in section 2.4, there have been no previous studies about this theme to the extent we have investigated. There are similar studies, but no matching ones. From the survey of prior studies, there is a suggestion that it may be possible to indicate the quality status by looking at the status of the issue session of the OSS project. Thus, we felt compelled to study this issue. This theme is studied in Chapter 4.

The third challenge in an enterprise software project willing to adopt OSS is that how to predict software quality. Since the user who decides to adopt OSS is responsible for the quality of OSS, prediction knowledge to predict the final software quality that can quantitatively judge whether or not it is available OSS is required. From the survey of prior studies, there is a suggestion that it may be possible to derive knowledge for predicting responsiveness and final quality by looking at the resolution rate and the status of speed and continuity of problem response in OSS projects. This theme is studied in Chapter 5.

The key contributions of this thesis are centered around improving the efficiency of corporate system software development in adopting OSS. We propose the following:

- A method of structured analysis of the evaluation process for adopting

OSS

- An OSS quality indication model of large-scale OSS projects for adoption decision-making
- An OSS quality prediction model considering support in use of large-scale OSS projects for adoption decision-making

3. Structured analysis of the evaluation process for adopting open-source software

In this chapter, we attempt to clarify the decision-making procedure using the structured map by extracting the evaluator layers, axes, and factors of OSS adoption evaluation.

3.1. Introduction

Recently, adopting open-source software (OSS) in the software development process, for reducing the development cost and development period, has become wide-spread among software development venders. It contributes to improving development effectiveness; however, it requires important factors to be considered such as it does relate to business alliance strategy, it does not insist on the monopolistic use of intellectual property [Chesbrough and Appleyard 2007], [JapanPatentAttorneysAssocitation 2006] and its does not guarantee its software quality, and so on. This implies that adopting OSS requires crucial decision making in terms of various aspects including business, technology, and intellectual property management, which are not mutually independent but may exhibit a complex set of relationships.

This chapter is organized as follows. In section 3.2, we briefly review how adopting OSS in the software development process is different from the conventional approach. In section 3.3, we propose a judgment factor and evaluation criteria axis when adopting OSS as the decision-making problem. Then, we consider a structured analysis by breaking down the evaluation axis and major factors. Finally, we note the structured evaluation criterion map and suggest possible directions for future investigations.

3.2. Adopting OSS in the software development process

Management of technology (MOT) maximizes the value creation of a company, in which input such as materials is converted into output such as customer value [Nobeoka 2006].

Here, we briefly review value creation in software development, as shown in Figure 2-1. For closed-source software development, which is performed within a company, a company's technology is the input, and the output is software to be used for business and accumulated as company assets. When OSS is adopted in software development, OSS, which is a third-party software, is also included as input to improve efficiency at the points of quality, development cost and period. As a side effect, there are limitations depending on the license of the OSS, such as disclosure of the source code, which affect the output.

Therefore, decision making in terms of various aspects when the OSS is introduced becomes important in earlier stages of the development process. Some of these aspects include effective development (e.g., cost, development period, quality, and operation), acquired technology (e.g., innovative functions and solutions in line with customer needs), and strategic business value (e.g., differentiation, originality, alliance, and barriers). Judgment must be made considering various aspects including business, product, technology, and intellectual property. It is noted that these judgment indexes are not mutually independent but may exhibit a complex set of relationships.

3.3. Structured analysis of the evaluation process when adopting OSS

3.3.1. Judgment Index and Evaluation Criteria Axis

For evaluating the introduction of OSS based on the roles and viewpoints of MOT [Nobeoka 2006] and the axis of the integrated roadmap of MOT [Tofu 2011], we will consider an OSS-specific point of view.

In terms of the roles and perspectives of MOT, the major indexes to be considered include technology, customer needs, competitive environment, differentiation and identity, enterprise business structure, and organizational structure. Technologies, products, and businesses are adopted as the axis of the integrated roadmap. From these points of view, MOT perspectives include the following as the evaluation criteria axis: (1) technology: includes differentiation and uniqueness; (2) products: include customer needs; (3) business projects: include competitive environment, differentiation, and distinctiveness; and (4) company operation.

Moreover, for an OSS-specific point of view, the following indexes should be investigated because OSS is third-party property as a deliverable by the developer community. In the case of adopting OSS, the trade-off between the creation of one's own property and the utilization of third-party property must be considered. Therefore, the indexes include (5) the creation of intellectual property and (6) the utilization of intellectual property. In terms of utilization, the items to be examined are, for example, the possibility of enclosing the technology of the deliverable and restrictions on owned or created patents by carefully examining the license of the OSS.

Alliance strategy is an OSS-specific evaluation sub-index, which is

categorized as a factor under several indexes. In recent years, there has been strategic OSS implementation by IT enterprises. Some of the developer groups are launched by enterprises. When OSS is introduced as a platform, applications and business models may be decided based on choices; therefore, it is also necessary to consider monetization by forming alliances considering network externalities. When combining one's own technology with an open technology, we clarify whether the purpose is supplementing or reinforcing one's technology and discuss its necessity. As the interface of joining is required to design with open and closed clarity, consideration of feasibility is necessary.

Value creation through "product" and "service" integration is another OSS-specific evaluation sub-index. Regarding the business model that connects a product to the Internet and adds value by web services, it is key to efficiently construct the communication section, which is a commonly used technology, and to focus on developing the core services of the distinctive business itself. There are many OSSs used for communication, and OSS coupling is important for the product emphasis on service. When services became more valuable, connecting functions became important, and OSS was used in large numbers, so the decision to introduce OSS became more important.

3.3.2. Evaluation Layers, Evaluation Criteria Axis and Factors

In this section, the evaluator layers, evaluation criteria axis, and factors are explained.

First, as the evaluation layers, the business executive layer, strategic planning layer, technological design layer, and software implementation layer are defined. Figure 3-2 shows the roles of each layer, corresponding to the V-shaped model of software development. The V-shaped model of software development is a definition of the design layer and the implementation layer, but here it is set as a plus two because it is necessary to judge the area of management and planning further upstream [IPA 2013] [IIBA 2015]. Table 3-1 summarizes the evaluation layer categories and their major evaluation factors.

The business executive layer analyzes the business environment in the first stage of development, ultra-upstream, and verifies the business results in the second stage. Prioritized factors to evaluate are investment policy, human resources, alliance possibility, and business structure,

The strategic planning layer plans the business strategy in the first stage of development, super upstream, and verifies the result of strategy implementation in the second stage. The main factors to evaluate are the company's strength (business and technology), positioning, technology innovation and maturity, and market requirements.

The technological design layer performs requirements analysis and basic design at the upstream of development and a system test and acceptance test at the latter stage. The main factors to evaluate are technological challenge, quality requirement, development cost and term, patent, and license.

The software Implementation layer is functionally and detailly designed, and then mounted and united/integrated. The main factors to evaluate are specification of interface and code, quality of code, maintenance, and difficulty

level of coding and testing.

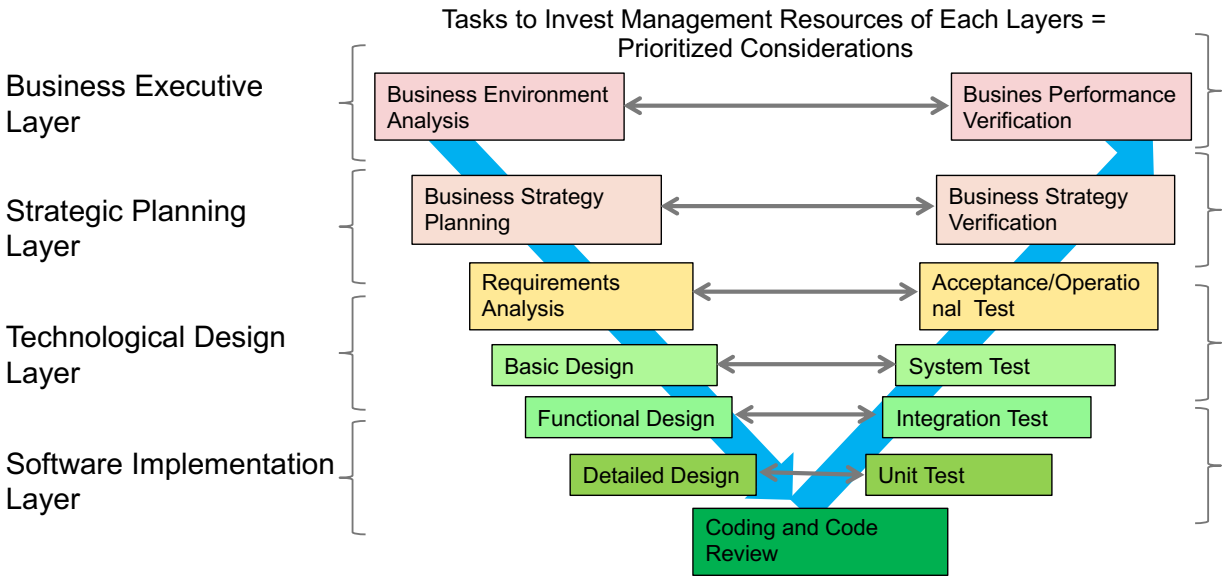

Figure 3-2. Evaluation Layer Category in adopting OSS

Table 3-1. Evaluation Layer Category and Major Factor in evaluation process when adopting OSS

Evaluator Layer Category	Major Evaluation Factors
Business Executive	Investment Policy, Human Resources, Alliance Possibility Business Structure, Engineering Resources
Strategic Planning	Strength (Business and Technology), Positioning, Technology Innovation and Maturity Market Requirements Characteristics of Customer/User Needs
Technological Design	Technological Challenge, Quality Requirement, Development Cost and Term, Patent, License
Software Implementation	Specification of Interface & Code, Quality of Code, Maintenance, Difficulty Level of Coding and Testing,

Next, the evaluation criteria axis and factors are defined.

The evaluation criteria axes are technology, products, business project, and company operation as four indexes of MOT and intellectual property creation and application as two OSS-specific indexes; these are identified in section 3.3.1. Regarding the sub-indexes in section 3.3.1, alliance strategy is defined as the factor of business project, and value creation through “product” and “service” integration is defined as the factor of products. Table 3-2 summarizes the evaluation criteria axis and their major evaluation factors.

Regarding the technology axis, each of the above-mentioned evaluation layers evaluates the strength of technology, technology innovation and maturity, quality of software and systems, engineering resource and so on. About the product axis, each layer evaluates customer needs, product/service integration as market requirements (integration of things), and so on. In terms of the business axis, each layer evaluates alliance strategies, competition circumstances, and investment for differentiation and uniqueness. As the management axis, each layer evaluates organization, human resources, business situation, financial situation and so on.

Regarding intellectual property creation, each layer evaluates novelty and inventive steps of the software created possibility of license-out and so on. In terms of intellectual property application, each layer evaluates license content of adopting software, availability of guard from the others, usage limitation of the patent, and so on.

Table 3-2. Evaluation Criteria Axis and Major Factor in evaluation process when adopting OSS

Evaluation Criteria Axis		Major Factors
MOT	Technology	Strength of Technology, Technology Innovation/ Maturity, Quality of Software and System, Engineering Resources
	Products	Customer Needs, Products/Services Integration
	Business Project	Alliance Strategy, Competition Circumstances, Investment for Differentiation/Uniqueness
	Company Operation	Organization, Human Resources, Business Situation, Financial Situation
OSS-Specific	Intellectual Property Creation	Novelty and Inventive Steps of the Software Created, Possibility of License-Out
	Intellectual Property Application	License Content of Adopting Software, Availability of Guard from the Others, Usage Limitation of the Patent

3.3.3. Structured Analysis

We consider OSS implementation as a decision-making problem. The final goal (problem) is "judgment of adopting OSS"; the evaluation criterion was set as the "evaluation factor" group extracted from the viewpoint of six evaluation axes defined in Table 1. We defined "alternative OSS solution" as "OSS introduction" and "company's development."

The evaluation factors were extracted by considering car navigation software development as an example. Car navigation systems have been developed not only for embedded software development but also for multifaceted applications because of the changes in market demand due to the spread of smartphones, the existence of OSS platform options such as Android, the intellectual property litigation risk of patent infringement and license violation, and technology commoditization. This is a case that requires decision making regarding to adopting OSS.

Figure 3-3 shows the overall hierarchical relationship of the evaluation layer, the evaluation criteria axis, and the major factors. Evaluators of four layers evaluate each factor in terms of six evaluation axes. Factors that require evaluation and unnecessary factors are different for each axis; it is assumed that one factor can be evaluated on multiple axes by changing the viewpoint.

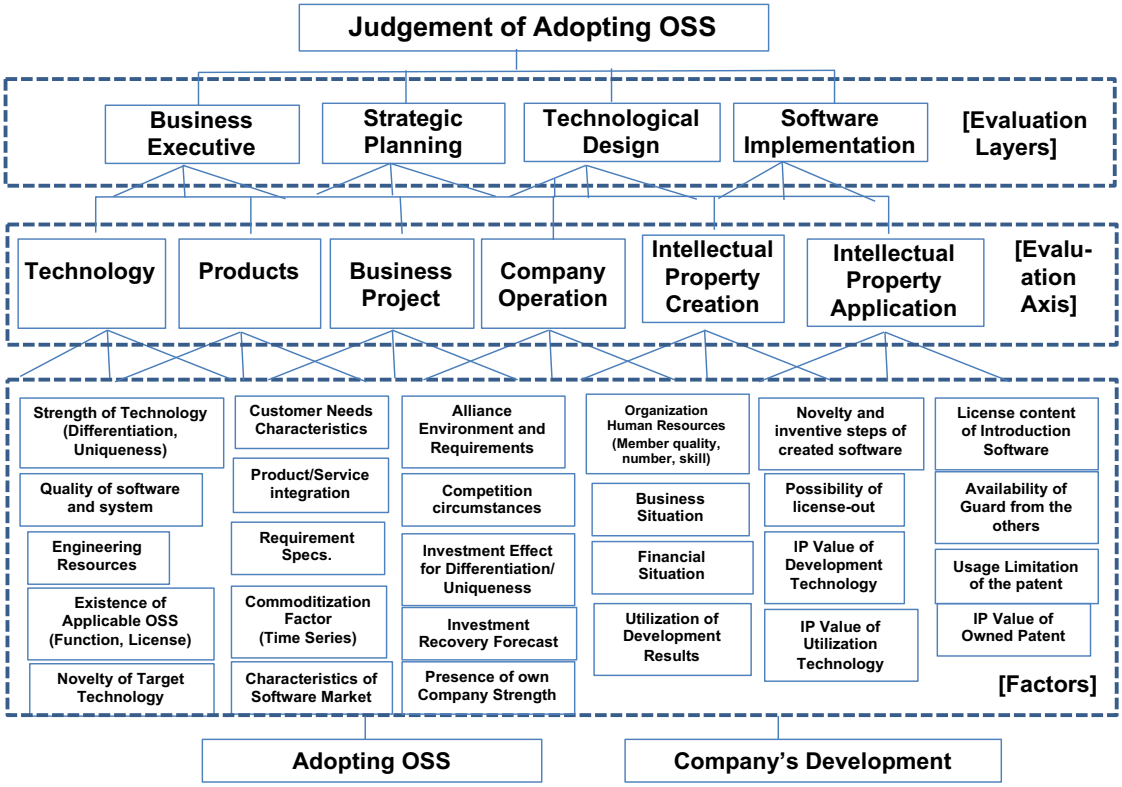

Figure 3-3. Overall hierarchical relationship of the evaluation layer, the evaluation criteria axis and major factor

In the following section, the relationship between the axis and major factors is analyzed. It is a hypothesis, and we will proceed with the verification of generalization in the future.

3.3.3.1 Technology Axis Factors

Figure 3-4 shows the result of examination of the technology axis factors. The factors related to the development period and cost are evaluated on the arc from the engineering resources to the development man-hours; factors related to the technical value of third-party technology introduction are evaluated next on the arc concerning the comparison of the difference between the company's technology strength and the development target; and then the delay in the commoditization of technology is evaluated to make the

decision of adopting OSS.

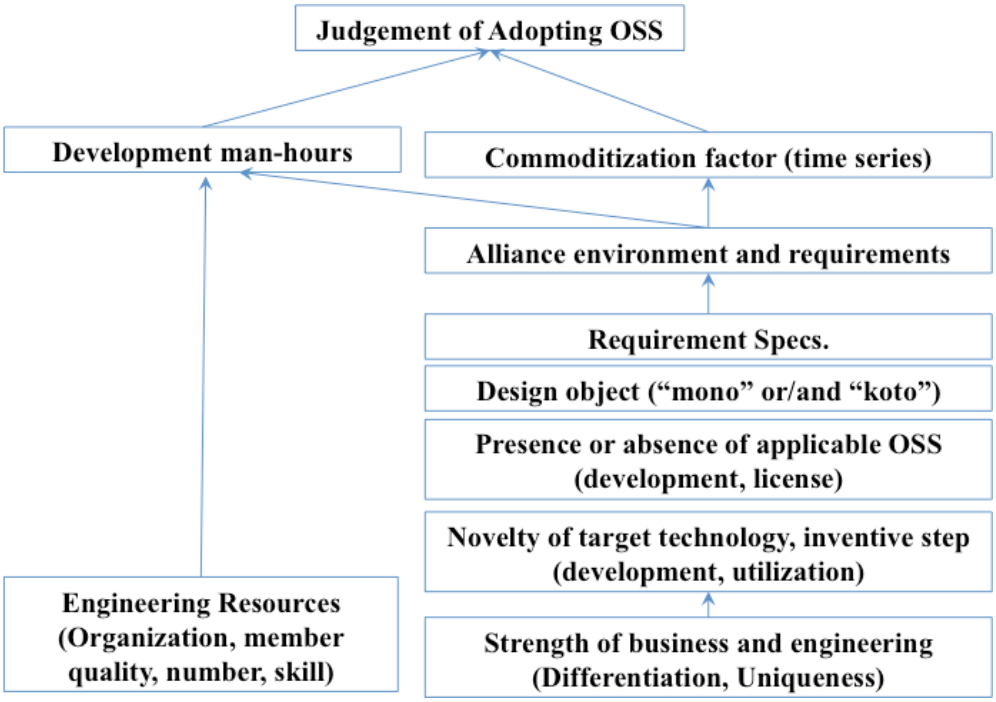

Figure 3-4. Hierarchical structure of technology axis factors

3.3.3.2 Product Axis Factors

Figure 3-5 shows the result of the examination of the product axis factors. From the comparison of the difference between the company' s strengths and customer needs and product/service integration requirement for the evaluation of the delay in the commoditization of products, we evaluate the effect of network externality of mono/koto' s applications and business models under alliance requirements to make the decision of adopting OSS.

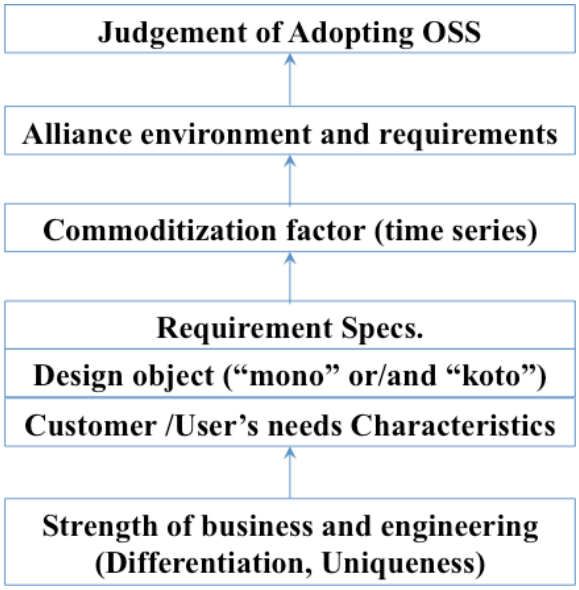

Figure 3-5. Hierarchical structure of products axis factors

3.3.3.3 Business Project Axis Factors

Figure 3-6 shows the results of the study on the business axis factors. There are arcs for evaluating customers, user characteristics, and required specifications and also for comparing intellectual property competitiveness to competitors; however, no hierarchical relationship among the major factors has been found.

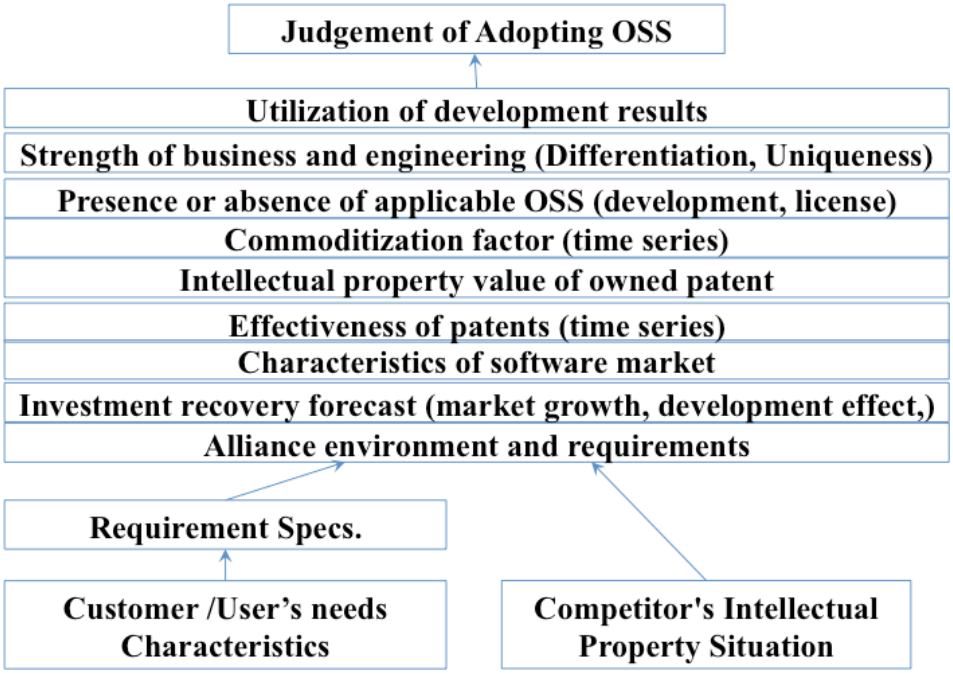

Figure 3-6. Hierarchical structure of business project axis factors

3.3.3.4 Company Operation Axis Factors

Figure 3-7 shows the results of investigations on the company operation axis factors. From the evaluation of engineering resources, there are arcs for the selection of the core/non-core business and evaluation of the investment scale; however, no hierarchical relationship among the major evaluation factors has been found.

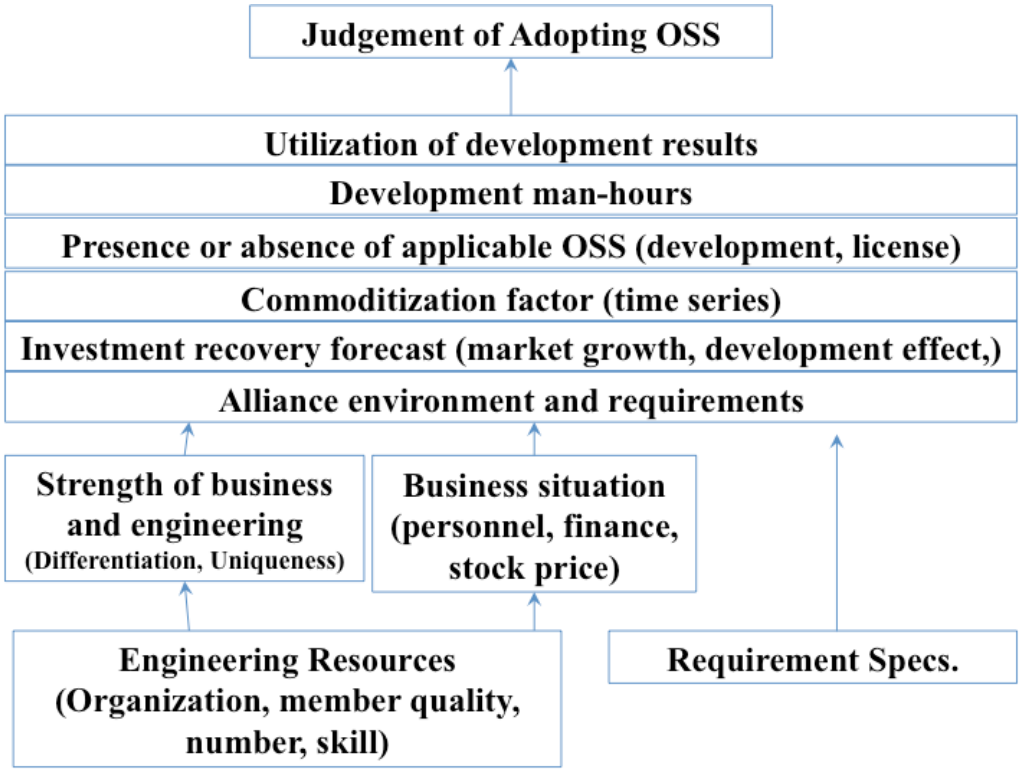

Figure 3-7. Hierarchical structure of company operation axis factors

3.3.3.5 Intellectual Property Creation Axis Factors

Figure 3-8 shows the result of the examination of the intellectual property creation axis factors. Although there are arcs from the evaluation of the novelty and inventive step of the creation to the evaluation of the innovativeness of the required specification and the arc of the evaluation of the intellectual property range that can be entitled to the design object (mono/koto), any major hierarchical relationship among the evaluation factors has not been found.

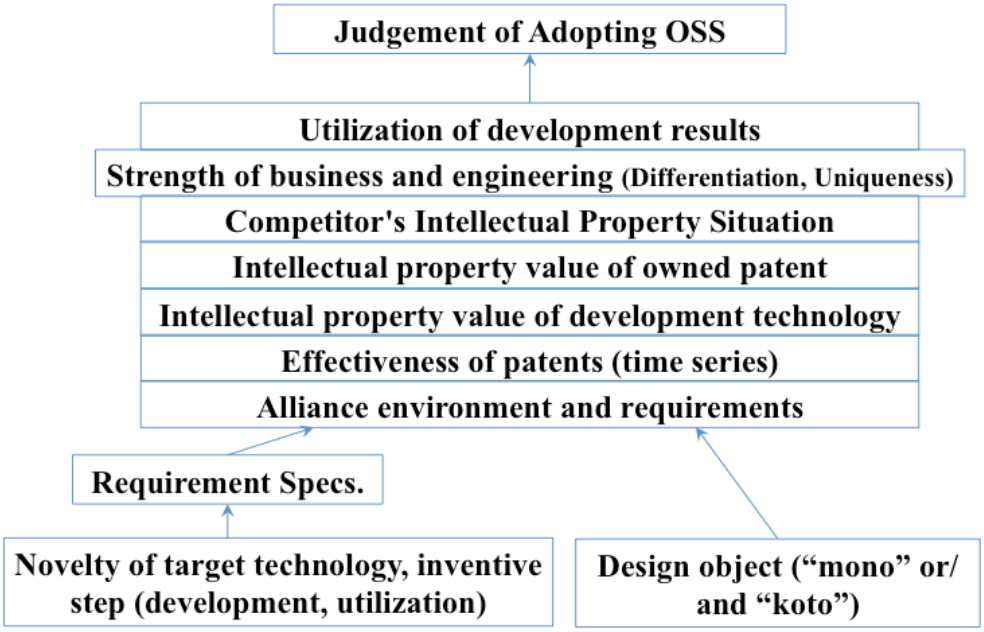

Figure 3-8. Hierarchical structure of intellectual property creation axis factors

3.3.3.6 Intellectual Property Application Axis Factors

Figure 3-9 shows the results of examination of the intellectual property application axis factors. The factors related to profit-and-loss arithmetic based on the intellectual value of OSS are evaluated; factors related to the alliance relationship and utilization of the developed output including OSS are evaluated to make the decision of adopting OSS.

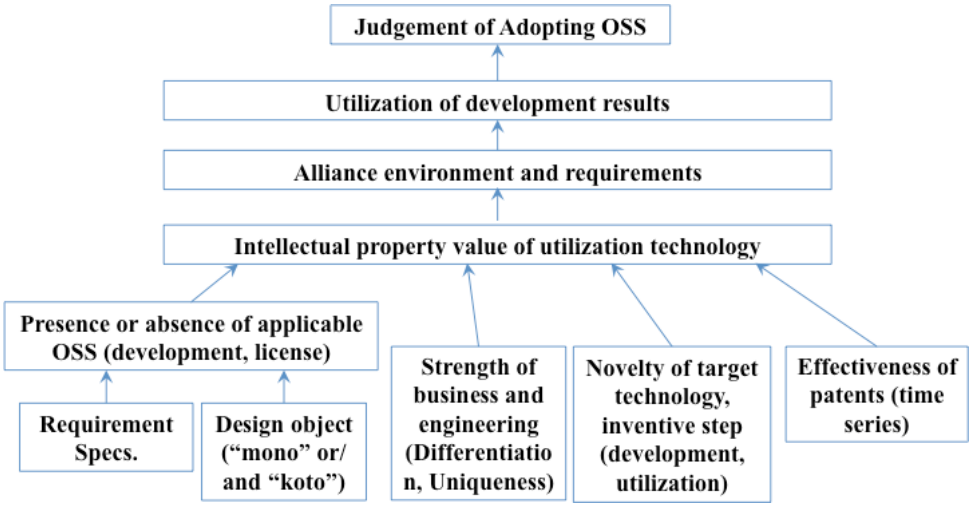

Figure 3-9. Hierarchical structure of intellectual property application axis factors

3.4. Structured evaluation criterion map for adopting OSS

We wanted to map the entire picture of this structure shown as Figure3-3, but it is impossible to represent it with a conventional model like the V-shaped model as in Figure3-2. This is because the two-axis model and V-shaped model are conventional waterfall-type development frameworks. Therefore, we decided to map the relationship between the evaluation factors for the six evaluation axes and the four evaluator layers on concentric circles. An example of the overall evaluation is shown in Figure 3-10.

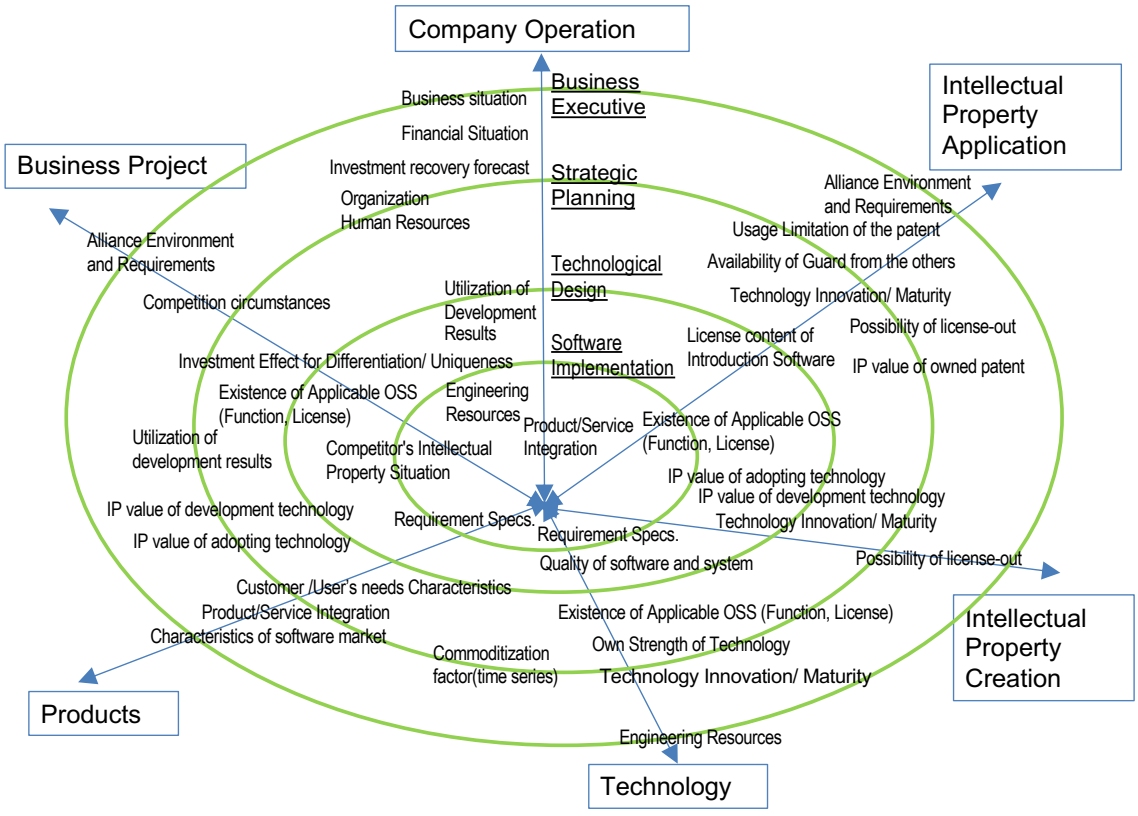

Figure 3-10. Example of the structured evaluation criterion map

The evaluation factors related to software implementation, technological design, strategic planning, and business executive are arranged in order from the criteria axis origin.

The usage example is as follows. Technology innovation/maturity is evaluated primarily by business executive layer and strategic planning layer from the perspectives of technology, business project, and intellectual property creation. Quality requirement and quality of software and system are evaluated primarily by the technological design layer and software implementation layer from the perspective of technology, product, and intellectual property utilization. Alliance environment and requirements are evaluated primarily by business executive and strategic planning from the perspectives of business project, technology, and intellectual property utilization.

It is presumed that this map will be created by the project manager responsible for the entire system and/or product based on discussions in meetings that transcend departmental boundaries. It is intended to be used by the project manager responsible for the entire system and/or product to make optimal OSS adoption decisions by giving a bird's-eye view of the overall status of the project.

3.5. Chapter Conclusion

For the decision making for adopting OSS, a method to analyze the relationship between the evaluation criteria axis and contributing factors was examined.

First, for the evaluation criteria axis, the basic axes from the MOT viewpoint were defined; then the axis of intellectual property creation and application were defined from the OSS-specific viewpoint to use third-party intellectual property. Second, the evaluation factors, using car navigation development as an example, were extracted. Each factor is not an independent event but is intertwined as a different real item (instance) on multiple axes, and we attempted to analyze the hierarchical structure of the factors for each axis.

The evaluation factors and axes of OSS adoption are influenced by the progress of the open innovation strategy and tactics of the company. The factors for each evaluation layers and axes will be examined continuously.

In this chaper, a method for organizing the evaluation axes and factors of OSS adoption evaluation and the structure of the evaluator hierarchy were studied. A structured map was proposed that defines the positioning of the OSS evaluated from the perspective of technology, products, business project, company operation, and intellectual property creation and application so that the project managers responsible for the entire system development can get an overview of the adoption evaluation.

4. Quality Indicator Model of Large-Scale Open Source Software Projects for Adoption Decision-making

In this chapter, we attempt to identify a quality indicator model to quickly determine whether the OSS under evaluation can be adopted before starting a detailed examination from the analysis of the activity of the OSS development projects by the OSS developer community.

4.1. Introduction

Open source software (OSS) has seen remarkable progress in recent years. Typical examples of OSS include the web service stack LAMP [Ware 2002] [Gerner et al. 2005], which is further composed of four OSS components, namely, Linux [Linux 2020], Apache [Apache 2020], MySQL [MySQL 2020], and PHP [PHP 2020] / Perl [Perl 2020] / Python [Python 2020]), and the Android [Android 2020] operating system. Furthermore, various OSS applications that have been developed for LAMP and Android environments have been made available for free, including their source codes. Moreover, OSS usage in corporate information systems has been essential.

Because the use of an OSS is typically free, and its source 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 [Yoshitaka et al. 2017].

Nevertheless, adopting OSS for software development processes improves development effectiveness and efficiency, but important factors need to be considered as OSS licenses do not promote the monopolistic use of intellectual

property. Particularly, 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 might instead 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 adopted development and attempted to clarify the structured evaluation criterion map [Akatsu et al. 2018].

In the practical consequences of the company, OSS adoption is decided based on the important priority of the projects concerning, for example, the function and quality and intellectual property of the desired OSS. However, if the positioning is the same, the OSS with better quality will be adopted, because the software quality of the system including OSS itself must be guaranteed by the company who adopted its OSS. From an OSS usage perspective, OSS quality includes the technical support ability in a broad sense. This is the reason to delve deeper into the quality aspect among the various indexes described in Chapter 3.

From a software engineering point of view, functionalities and quality are the two important decision factors for adoption. Regarding OSS, the functionalities could be verified by surveying the source code due to the OSS's remarkable characteristic of delivering this code freely. In contrast, as the deliverables of an OSS are developed by its developer community, their quality is not guaranteed. The user corporate that decides to adopt the OSS is responsible for evaluating the quality of the desired OSS.

One index for measuring the quality is how the desired software is extensively used. Regarding ES, which is a software product of enterprise, it can be assumed that the proven software is of good quality. However,

regarding OSS, the track record of adoption is not disclosed. On the other hand, the activity status of the OSS developer community, including specification creation and code implementation and testing, is disclosed at development community websites such as GitHub. Appropriate maintenance by OSS developers is important, so the definition of OSS quality can be considered how sufficiently it is maintained by OSS development communities.

Thus, before examining the detailed deliverables of an OSS thoroughly, it is required to have knowledge about OSS quality, i.e., indicators for decision, that corporate business could use for quantitatively assessing whether a desired OSS should be adopted. The objective of this study is to propose an OSS quality indicator model and assessment method by examining the status of both created and closed user-raised issue sessions in OSS development projects.

4.2. Method for Deriving OSS Quality Indicator Model

Thus, in this study, the quality of an OSS is defined based on the manner in which software code and specifications are maintained by the 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.

Despite not solely being intended for the quality-related discussions, sessions at the OSS developer community website are crucial for building a quality indicator model, considering the following two factors. Firstly, the session contains a certain proportion of quality-related content, including bug reports and specification improve requests. Second, fast response and resolution among OSS developer/user community are indispensable in order to maintain the acclaimed quality of OSS.

Therefore, considering the first factor, in order to evaluate the trends for the amount of quality handled, we examine how the trends for changes in the cumulative number of created and closed issue sessions is shifted. Regarding the second factor, in order to estimate the quickness and continuity of the quality handled, we observe when the number of created and closed sessions becomes dissociated.

Consequently, we built a quantitative model regarding OSS quality applicable as knowledge by observing the status of sessions based upon two axes, which are the trends for transition and timing of deviation between created and closed issue sessions.

4.2.1 Axis 1: Trends for transition of cumulative number of created/closed

issues

The trends are summarized in three categories, “Linear,” “Logarithmic curve,” and “Cubic curve” for the following reason: Linear is supposedly a general transition, in other words, monotonically increasing. Logarithmic curve shows that session's created/closed number has been gradually increasing in the latest phase, then we could assume that it is in a stable state period. By contrast, Cubic curve shows that session's created/closed number is increasing in the latest phase, meaning that there is a “potential” which the quality information is increasing.

The method for classifying trends as “Linear,” “Logarithm,” or “Cubic” is as follows: First, curve fitting with linear and nonlinear regression is performed toward the time-series data plots of a cumulative number of created and closed issues regarding each project. Nonlinear regression here uses logarithmic functions and cubic functions. Second, whether the classification target of the OSS project is classified as Linear—where R2, the determination coefficient, is large enough—is examined. The reason to classify firstly as Linear is that it is a general transition. Then, those deviating from a Linear definition are assessed for Logarithm or Cubic definition. Finally, the judgement of Logarithm or Cubic is classified by which the R2, coefficient of determination, of both are relatively smaller.

4.2.2 Axis 2: Timing of deviation between the number of created and closed issues

The deviation timing between the number of created and closed issues is summarized in three categories: “Early stage,” “Middle stage,” and “Late stage.”

In this study, we define the deviation timing as the points at which the difference between the created number and the closed number exceeds 1% of

the created number. The difference is calculated by taking a 3-month moving average. If there is no divergence, it can be concluded that quick response and resolution have been achieved. Next, to classify the timing of the deviation, we divide the period between the project start and the time of data acquisition (April 22, 2020, for this study) into three parts. The period between the start and 33% is the Early stage, from 34% to 66% is the Middle stage, and from 67% to the data collection is the Late stage.

The method for classifying the “Early,” “Middle,” and “Late” stages is the following: By searching the number of created and closed cases for each month, the deviation timing is identified for each project. Then, the corresponding stage is assigned according to the percentage of the project period.

4.3. Evaluation of the Model

We attempt to improve the understanding by examining the 39 large-scale OSS projects according to the method described in Chapter 2. These data of 39 projects are the research materials that we use in various studies, so they are the universal datasets within our research group.

4.3.1. Target Project Selection

To analyze OSS quality, it is necessary to obtain a large amount of OSS development data. The source of the analysis data may be GitHub [GitHub 2020b] or Bitbucket [Bitbucket 2020], which are web sharing services providing a version control system. In this study, we used OSS development project data published on GitHub.

GitHub uses the version management system Git [Git 2020] and provides the web application program interface (API) GitHub API v3 [GitHubAPI 2020], with which users can access repositories and directories storing project deliverables and development history, among others. In this study, we used the GitHub REST API to select the target projects and examine them. For example, the function "Issues"—the issue management feature in GitHub—returns the timestamp for the issue creation date, the status of the issue, and its related comments.

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
- Extract projects whose repository size is 15 MB or more to ensure a large-scale OSS is selected

- 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 and commit numbers are within the second quartile or over to avoid bias due to 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 with missing values and projects wherein no deviation was discovered. Finally, 39 projects were selected.

These selected sample projects are listed in Table 4-1. In particular, Table 4-1 shows the project data as of April 22, 2020, which was extracted from GitHub on April 22, 2020. In Table 4-1, 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 660,000, which is sufficient for statistical analysis.

Table 4-1. Number of created issues, closed issues, and resolution rate for the 39 selected projects as on April 22, 2020.

No.	Repository Name	Created Issues	Closed Issues	Resolution Rate
1	alluxio	11,319	11,011	0.972789116
2	ansible	68,889	62,867	0.912584012
3	atom	20,277	19,586	0.965921981

4	bokeh	9,948	9,436	0.948532368
5	bosh	2,257	2,127	0.942401418
6	canjs	5,475	5,171	0.944474886
7	Cataclysm-DDA	39,786	38,650	0.971447243
8	collectd	3,442	2,979	0.865485183
9	conda	9,851	8,315	0.844076743
10	contiki	2,645	2,046	0.773534972
11	core	37,254	35,775	0.960299565
12	crystal	9,032	8,012	0.887068202
13	darktable	4,788	4,294	0.896825397
14	DefinitelyTyped	44,085	40,693	0.923057729
15	django	12,751	12,522	0.982040624
16	druid	9,718	8,720	0.897303972
17	Firmware	14,717	14,039	0.953930828
18	habitica	12,097	11,785	0.974208481
19	hazelcast	16,900	16,148	0.955502959
20	kotlin	3,303	3,081	0.932788374
21	libgdx	5,995	5,633	0.939616347
22	linux	3,555	3,247	0.913361463
23	llvm	69	67	0.971014493
24	mpv	7,588	7,147	0.941881919
25	neo4j	12,448	12,238	0.98312982
26	nixpkgs	85,666	79,911	0.932820489
27	opencv	17,094	15,347	0.897800398
28	phpmyadmin	16,058	15,535	0.967430564
29	ppsspp	12,720	11,817	0.929009434
30	PrestaShop	18,714	16,869	0.901410709
31	presto	14,415	13,426	0.931390912
32	radare2	16,664	15,310	0.918747
33	ReactiveCocoa	3,690	3,641	0.986720867
34	rethinkdb	6,831	5,424	0.794027229
35	RIOT	13,917	13,145	0.944528275
36	servo	26,233	23,082	0.879884115
37	spring-boot	20,955	20,479	0.977284658
38	web-platform-tests	23,119	21,343	0.923180068
39	yii2	17,634	17,117	0.970681638
	Total	661,899	618,035	

4.3.2. Trends for Change in the Number of Created and Closed Issues

In order to analyze the overall issue resolution rate for the 39 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 4-1. The blue line is the plots for created, and the red line is the plots for closed issues. The red dotted vertical line shows the deviation timing.

First, the corresponding relationships between the increase in the number of created issues and their resolution are presented in Figures 4-1(a)–(i); these relationships are as follows:

- Figure 4-1(a), (b), (c): Numbers of created and closed issues continuously match each other.
- Figure 4-1(d), (e), (f): Numbers of created and closed issues diverge during the middle stage of development.
- Figure 4-1(g), (h), (i) Numbers of created and closed issues are divergent from the beginning.

Second, three patterns can be recognized from the increasing number of created issues, also in Figure 4-1(a) –(i); these patterns are as follows:

- Figure 4-1 (a), (d), (g): Increases 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.
- Figure 4-1 (b), (e), (h): Increase in the number of issues was large at the beginning but decreased at the end, i.e., the number of issues increased

logarithmically.

- Figure 4-1 (c), (f), (i): Increase in the created issues is small in the initial stage, large in the middle stage, and small again in the final stage, i.e., the increasing trend follows the curve for the third power.

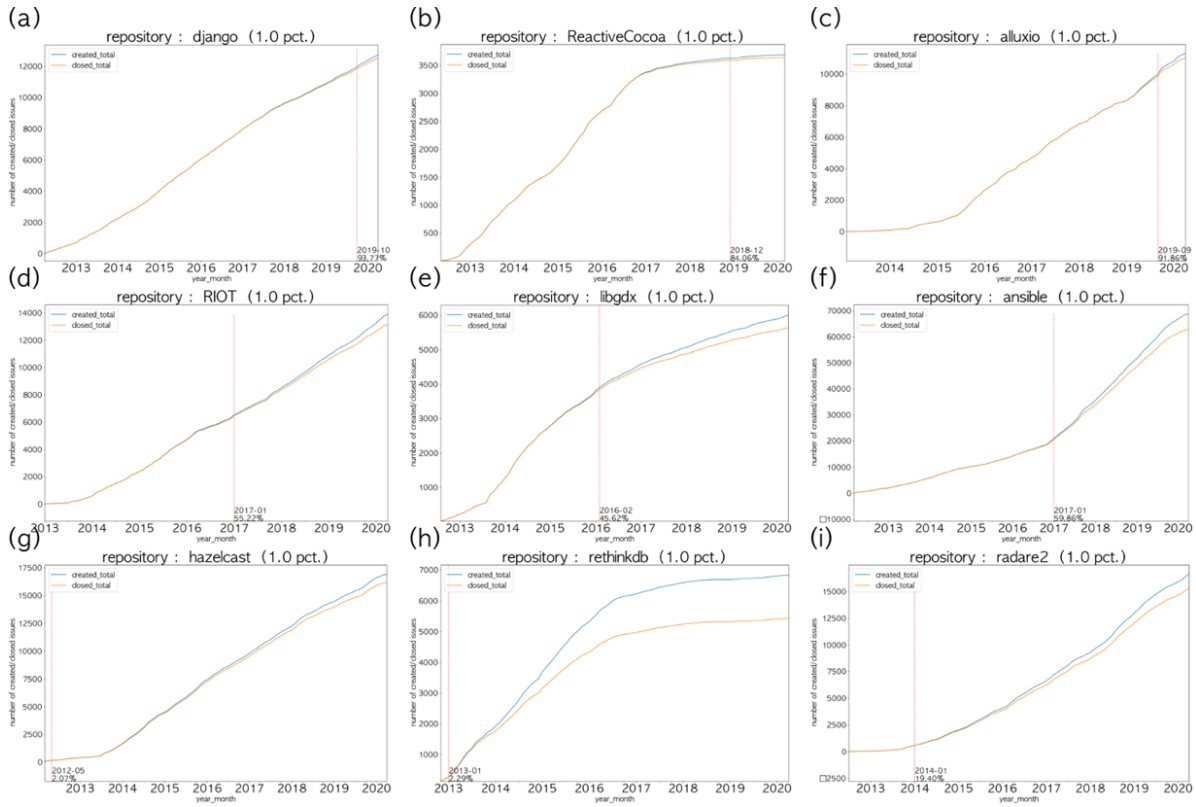

Figure 4-1. Examples of trends for changes in cumulative number of created and closed issues

4.3.3. Evaluation of the Result of Method Axis 1

The objective of Method Axis 1 is to classify the 39 selected projects as either "Linear," "Logarithm," or "Cubic." The results for the Linear category are shown in Figure 4-2, and for the Logarithm and the Cubic in Figure 4-3.

We examine the plots of the increasing number of created issues for all

projects. Then, we choose the threshold value of R2, the determination coefficient, as 0.975. In total, 16 projects, whose R2s are greater than 0.975, are classified as Linear.

We examine the plots of the increasing number of created issues of the projects leaks from Linear. Then, we compare the value of R2, the determination coefficient, of Logarithmic and Cubic functions. In total, 9 projects are classified as Logarithm, and 14 projects as Cubic.

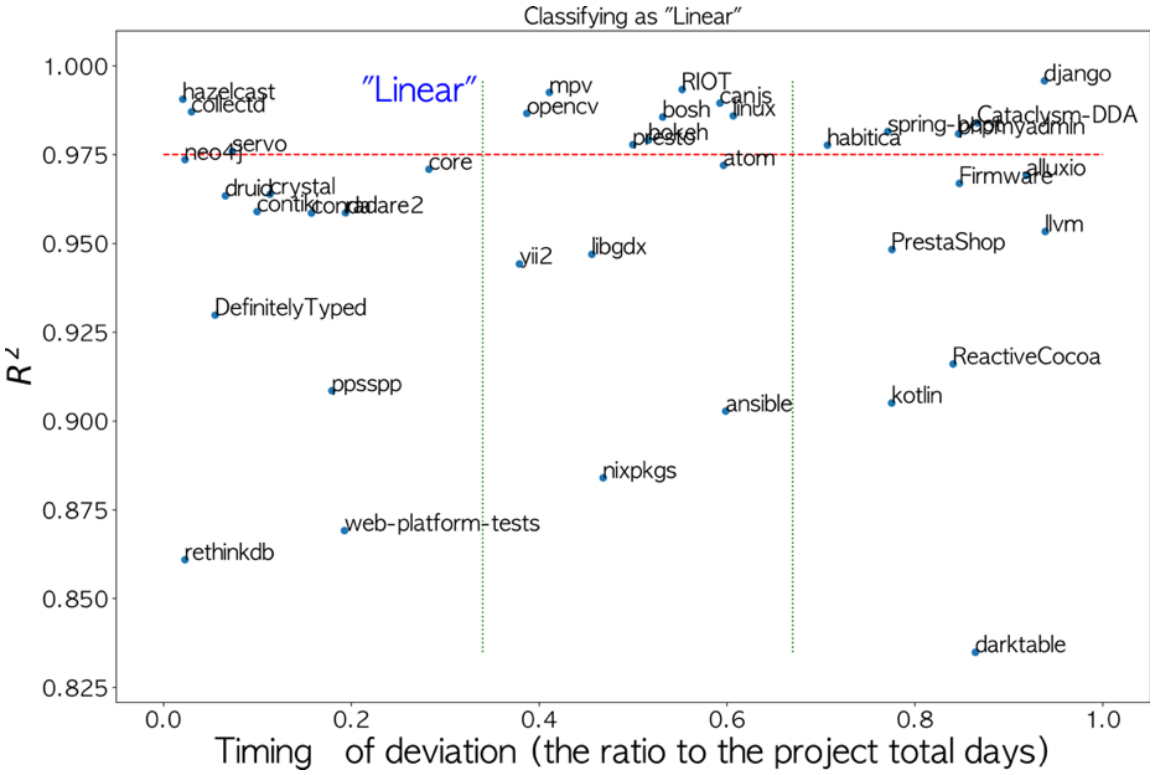

Figure 4-2. Result of the analysis for the “Linear” category.

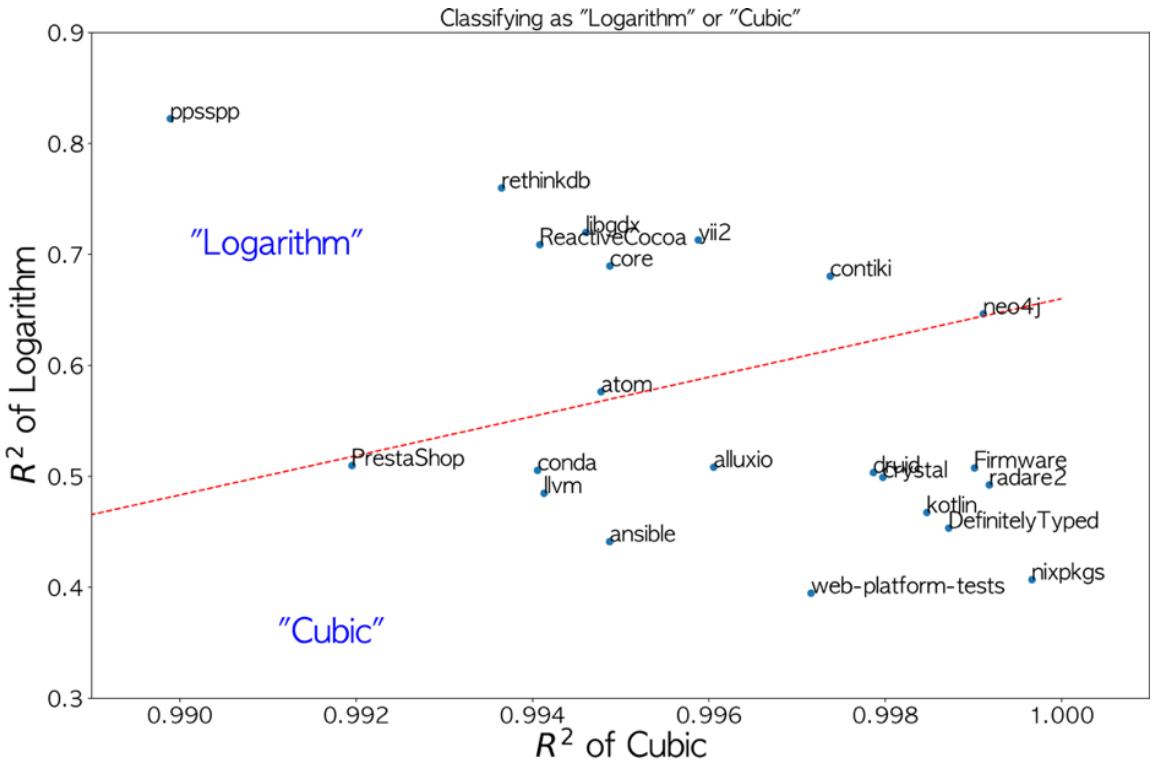

Figure 4-3. Result of the analysis for the “Logarithm” and the “Cubic” categories.

4.3.4. Evaluation of the Result of Method Axis 2

The objective of Method Axis 2 is to classify the 39 selected projects in the “Early stage,” “Middle stage,” or “Late stage” categories. Plots of three kinds of timing are shown in Figure 4-4. The period between the project start and the time of data acquisition (April 22, 2020, for this study) is divided into three parts. The period between the start and 33% is the Early stage, from 34% to 66% is the Middle stage, and from 67% to the data collection is the Late stage.

The deviation timing depends on the projects. Three stages classification seems reasonable because there is a tendency that projects deviated lately make better final resolution rates.

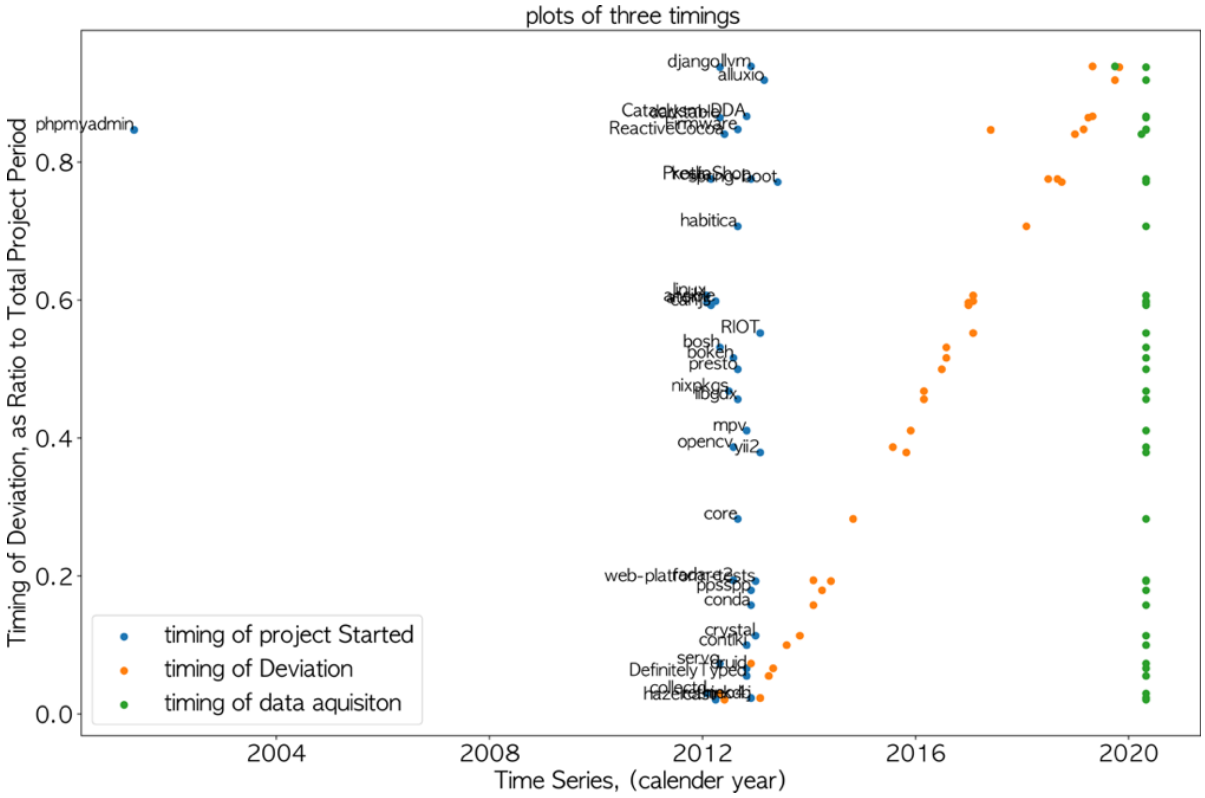

Figure 4-4. Plots of the timing of project started, deviation, and data acquisition

4.4. Discussion

As a summary of the evaluation, the results shown in Figure 4-5 are obtained. The horizontal axis is issue increase trends, which is divided into a linear curve, a logarithmic curve, and a cubic curve, respectively. The vertical axis is the timing of unresolved issue occurrences, which is divided into late, middle, early, respectively. 39 projects are mapped to this nine-quadrant matrix. The three "Late" regions in the timing of unresolved issue occurrences and the three "Logarithm" regions in the issue increase trends are labeled blue "Convergence Continues" and "Stable State," respectively. It is the T-shaped area of the alphabet.

		Increasing Trend of Issue Creation		
		Linear	Logarithm	Cubic
Timing of Unresolved Issue Occurences	Late	['Cataclysm-DDA', 'django', 'habitica', 'phpmyadmin', 'spring-boot'] ***Convergence continues***	['ReactiveCocoa'] ***Convergence continues Stable State***	['alluxio', 'darktable', 'Firmware', 'kotlin', 'llvm', 'PrestaShop'] ***Convergence continues***
	Mid	['bokeh', 'bosh', 'canjs', 'linux', 'mpv', 'opencv', 'presto', 'RIOT']	['atom', 'libgdx', 'yii2'] ***Stable State***	['ansible', 'nixpkgs']
	Early	['collectd', 'hazelcast', 'servo']	['contiki', 'core', 'neo4j', 'ppsspp', 'rethinkdb'] ***Stable State***	['conda', 'crystal', 'DefinitelyTyped', 'druid', 'radare2', 'web-platform-tests']

Figure 4-5. T-model: Nine-quadrant Matrix consisting of Issue Increase Trends and Timing of Unresolved Issue Occurrences

Considering the meaning of the axes of the T model, it can be interpreted that the axis of issue increase trends indicates "the technological maturity,"

and the axis of the timing of unresolved issue occurrence indicates “the activeness of the development community.” As technological maturity increases, the tendency for issue occurrence is expected to diminish. In the case that the activeness of the development community is high, the tendency for unresolved issue occurrence is expected to be low. Recognizing that the “logarithm” type has a high technology maturity level, we interpreted its increase as caused by the issues of unfatal matter such as comments rather than fatal problems. Detailed content analysis will be reserved for a future study.

Assessing only these quantitative evaluations, we can see that the projects in the part composed of the “Late” row and the “Logarithm” column (where “Convergence Continues” and “Stable State” are shown in blue) maintain the adequate quality that can be sufficiently used for system development in business corporate.

Therefore, we name T-model the quality indicator model for OSS adoption decision-making from the form shown in Figure 4-5. It is derived that these two axes, “the technological maturity” and “the activeness of the development community” are important indicators for determining the quality of OSS.

This T-model was evaluated as a result without any discomfort by four software development engineer specialists.

Regarding the OSS project revealed from this T-model region, it is necessary to analyze the availability with additional investigation. The method could be the way as it has been presently. We found a possible indicator, explained briefly below, during the examination.

First, we investigated 19 projects revealed from the T-model region. Figure

4-6 shows that the ratio of the pro-deviation and post-deviation periods and the final resolution rate correlate. Second, we investigated 11 projects in the Linear category, revealed from the T-model region. Figure 4-7 shows that the difference between the increase slope of created and closed issues and the final resolution rate are correlated. They will be further studied.

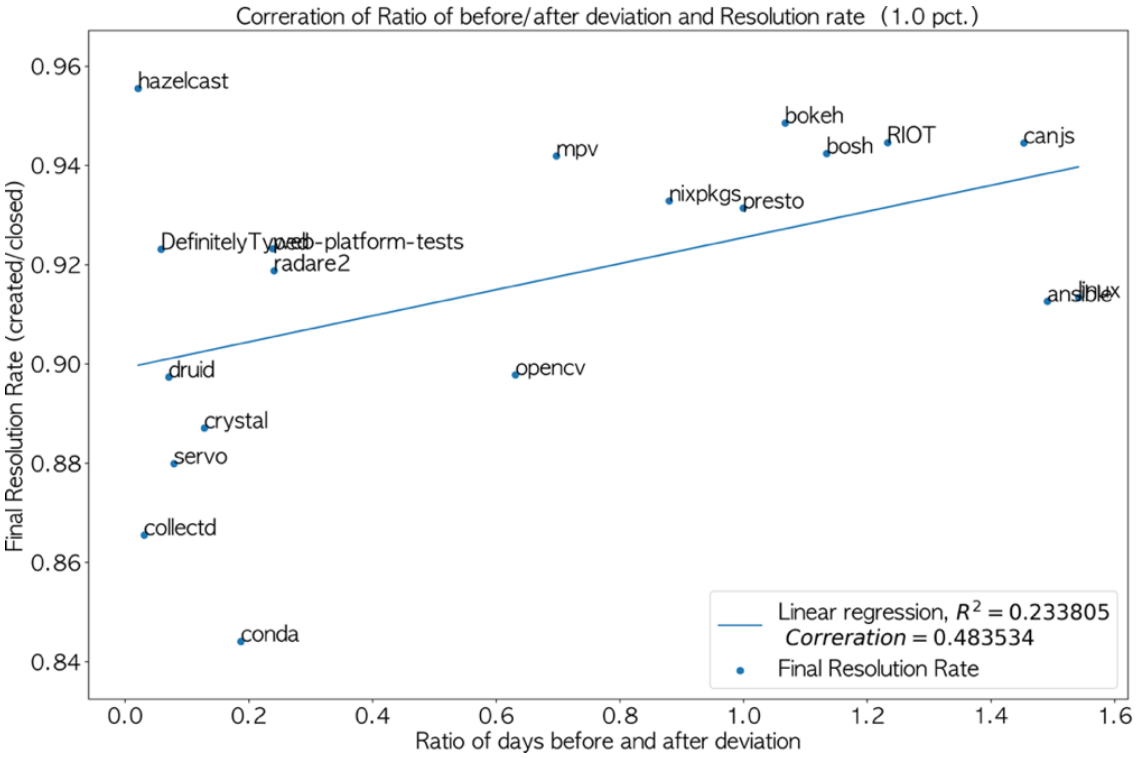

Figure 4-6. Correlation between ratio of pro-deviation and post-deviation period and Final Resolution Rate

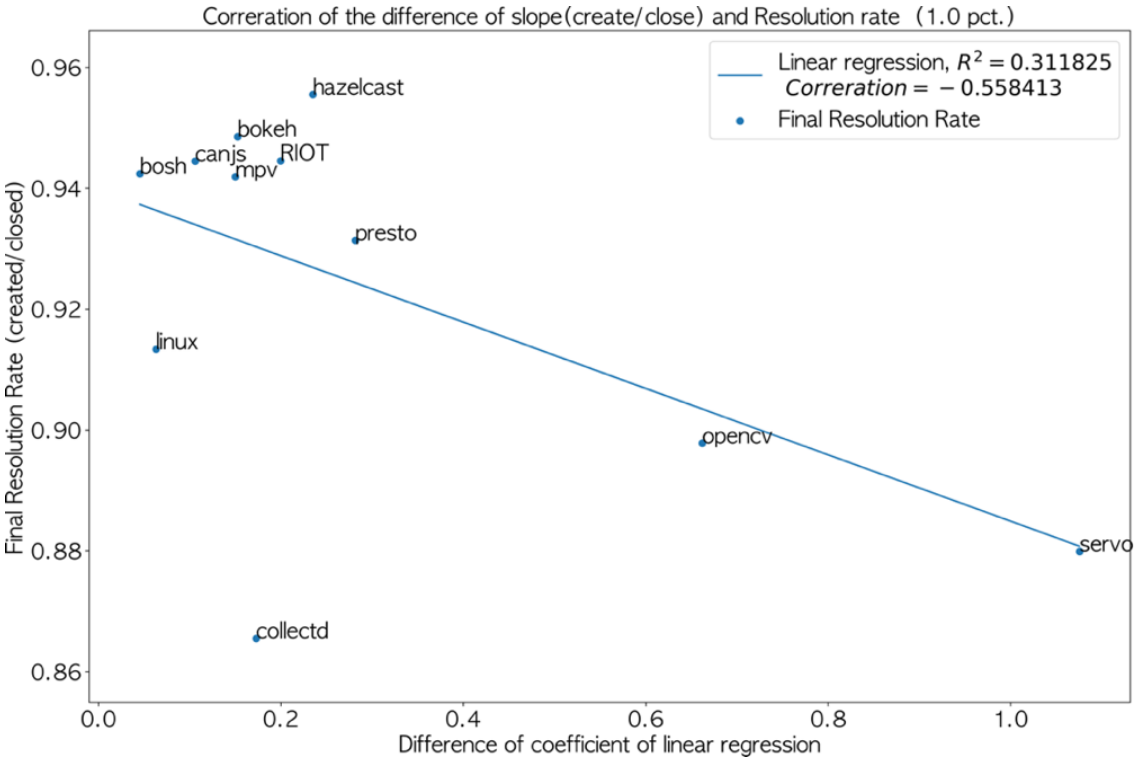

Figure 4-7. Correlation between the difference of increase slope and Final Resolution Rate

4.5. Chapter Conclusion

The T-model proposed in this paper has shown the possibility of speedy decision-making regarding whether to adopt the desired OSS for software development of a corporate information system. This T-model simply indicates “a region where there is generally no quality problem.” It does not indicate that the OSS project leaks from this region are “unavailable” for the target system development.

Regarding the OSS project revealed from this T-model region, it is only necessary to assess the availability as it has been presently by comparing with other various indexes, such as the degree of matching with the specifications in system development, required quality, required delivery time, and skills of the development team.

It is proposed to utilize the T-model as an index to quickly judge whether the target OSS can be used before starting a detailed examination.

The T-model consists of two axes: the maturity of the technological innovation and the activeness of the developer community. It was derived that the maturity and the activeness are important indicators for determining OSS quality. As a result of the analysis of these two axes, it was found that more accurate quality information may be obtained if a more detailed analysis is performed on each axis.

5. Quality Prediction for Large-Scale Open Source Software Projects

In Chapter 4, it was derived that the maturity of the technological innovation and the activeness of the developer community are important indicators for judging OSS quality. Of these two indexes, the technological maturity of the desired OSS could be evaluated explicitly by the development period and its usage frequency. Therefore, in this chapter, we delve deeper into the activeness of the OSS developer community.

5.1. Background and Purpose of This Chapter

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 [Ware 2002] [Gerner et al. 2005]—which, in turn, is composed of four OSS components namely Linux [Linux 2020], Apache [Apache 2020], MySQL [MySQL 2020], and PHP [PHP 2020] / Perl [Perl 2020] / Python [Python 2020]—and the Android [Android 2020] 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 the 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 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 [Yoshitaka et al. 2017].

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-adopted software development and attempted to clarify the structured evaluation criterion map [Akatsu et al. 2018].

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 [NationalTaxAgency 2020]. 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 report bugs. However, there are cases in which 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 for 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 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 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 [Yoshitaka et al. 2017] [Akatsu et al. 2018] [National Tax Agency 2020] [Krogh et al. 2012] [Izquierdo et al. 2015] and software quality [Tosun et al. 2010] [Radlinski 2011] [Xing et al. 2005]. However, regarding the quality of OSS, prior studies

are limited to quality processes on the development side [Bahamdain 2015], while there has been no discussion of predicting quality from the perspective of the user, except for a recent study by our research group [Akatsu et al. 2020].

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 [GitHub 2020b]. 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.

5.2. Definition of OSS Quality and its Measurement

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

5.2.1 OSS Quality

A 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 regarding 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 the 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.

5.2.2 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 using their issue management data.

Here, resolved means that the created issue has been closed, and unresolved

means that the created issue has not been closed. The resolution rate is the quotient of the number of closed issues divided by the number of created issues.

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.

5.3. 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.

5.3.1 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 [GitHub 2020b] or Bitbucket [Bitbucket 2020], 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 [Git 2020] and provides the web application program interface (API) GitHub API v3 [GitHubAPI 2020], 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.

5.3.2 Target Project Selection

In this study, sample projects were selected using GitHub API v3 [GitHubAPI 2020] 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 were long-term data.

- Extract projects whose repository size is 15 MB or more to ensure a large-scale OSS was selected.

As a result of surveying companies in terms of the OSS repository size (≥10 MB, ≥15 MB, or ≥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 [Charette 2009], which is about 10 times that of a 15 MB repository. In a previous study [Masuda et al. 2019], 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 5-1. In particular, Table 5-1 shows the project data as of December 31, 2018, which were extracted from GitHub on April 23, 2020. Data as of December 31, 2018, were selected in order to eliminate the influence of unresolved issues between December 31, 2018 and March 31, 2020 on our analysis. In Table 5-1, 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.

Table 5-1. 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	

5.3.3 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 5-1.

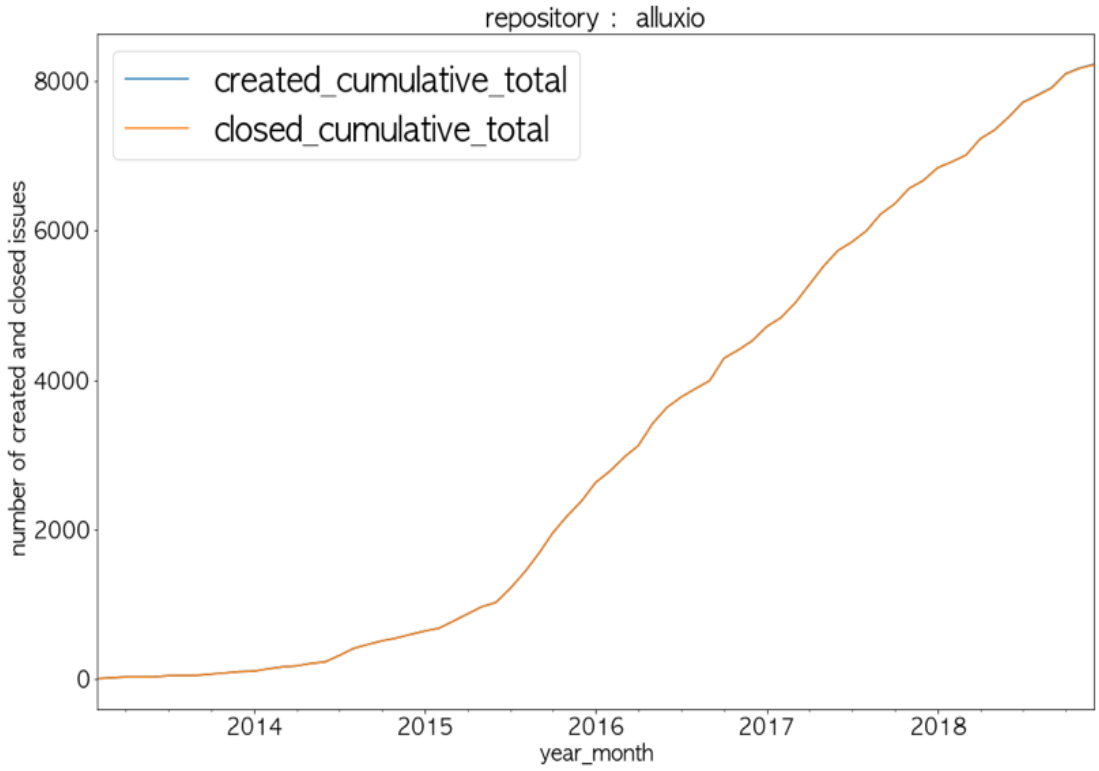

Figure 5-1(a). Examples of trends for changes in cumulative number of created and closed issues. (a)Numbers of created and closed issues continuously match each other.

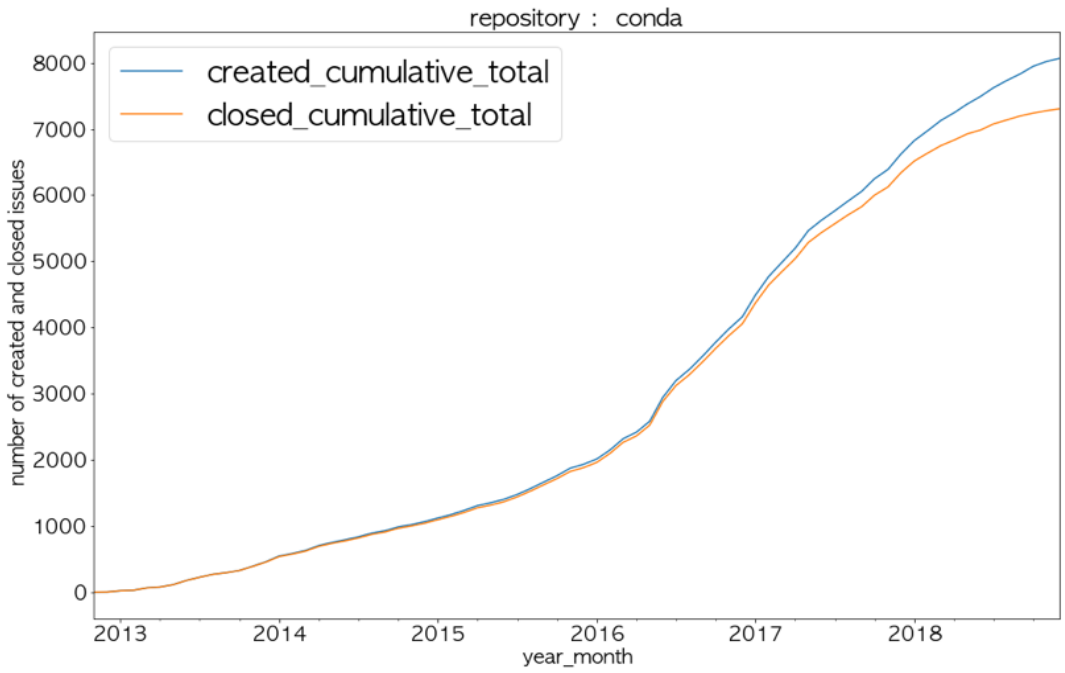

Figure 5-1(b). Examples of trends for changes in cumulative number of created and closed issues. (b) Numbers of created and closed issues diverge during the middle stage of development.

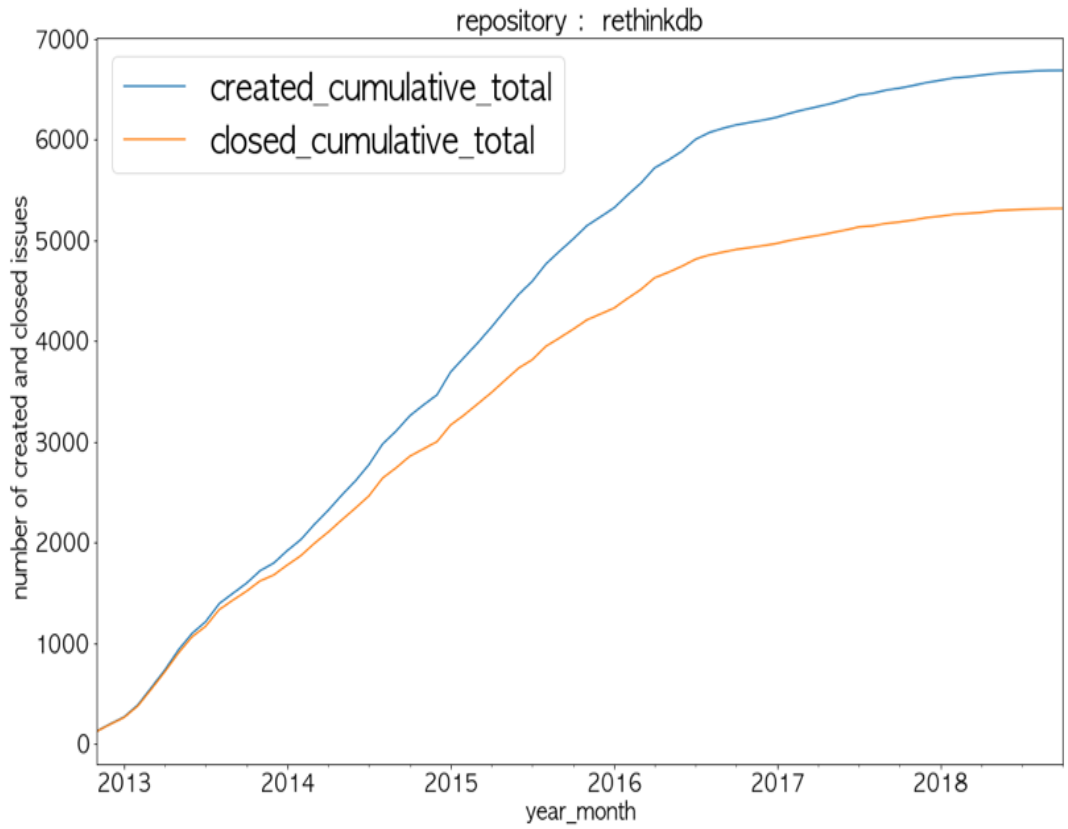

Figure 5-1(c). Examples of trends for changes in cumulative number of created and closed issues. (c) Numbers of created and closed issues are divergent from the beginning.

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 5-1(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 5-1(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.

Moreover, as shown in the example in Figure 5-1(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 5-1. 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.

Second, the corresponding relationships between the increase in the number of created issues and their resolution are also depicted in Figures 5-1(a)–(c); these relationships are as follows:

- Figure 5-1(a): Increases in the number of created and closed issues are aligned together.
- Figure 5-1(b): Increases in the number of created and closed issues begin to diverge during the middle stage.
- Figure 5-1(c): Increases in the number of created and closed issues remained separate from the beginning.

In Figure 5-1(a), a pattern is observed wherein new issues are continuously generated and also continuously solved; consequently, the final resolution rate is high. Figure 5-1(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 5-1(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 would 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.

5.4. 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 these data to derive knowledge that can be applied to the proposed AI-based quality prediction model.

5.4.1 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 5-2. In short, there is more the “dark blue,” which means “closed in 1 month” in the graph, the more prompt issue resolution is realized. The following observations were made for the three examples discussed in Clause 5.3.3.

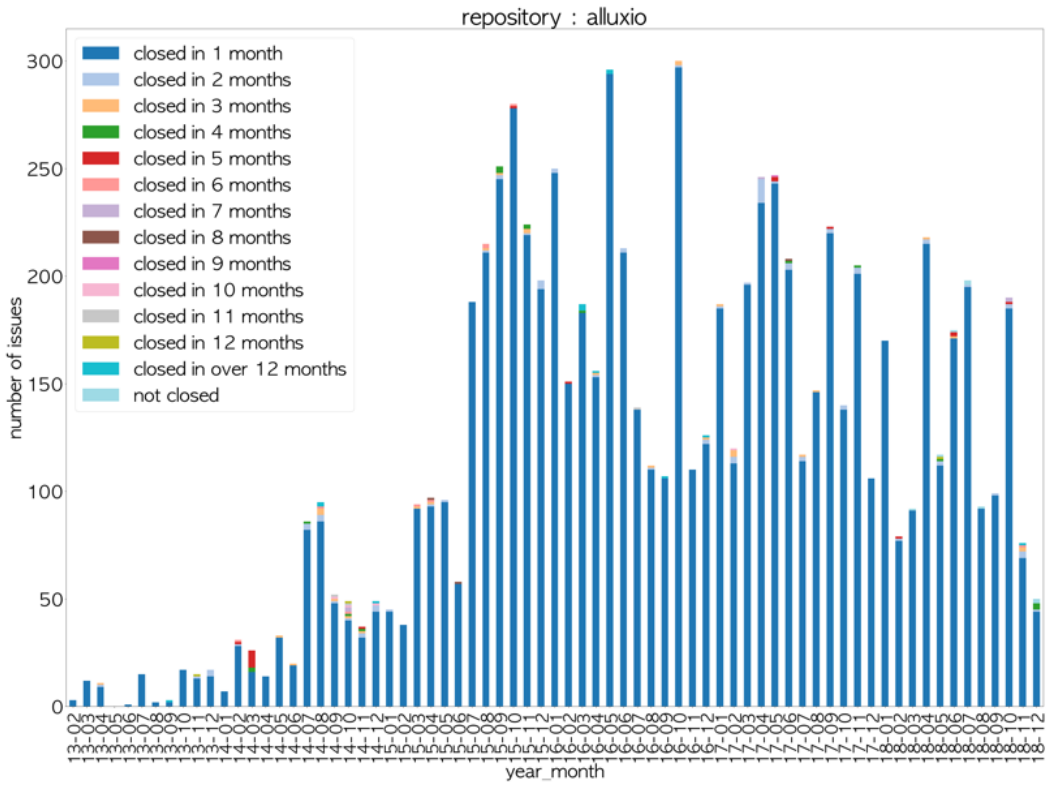

Figure 5-2(a-1). Patterns of transition in resolution status for each month by period. (a-1) Created issues continue to be resolved in the current month.

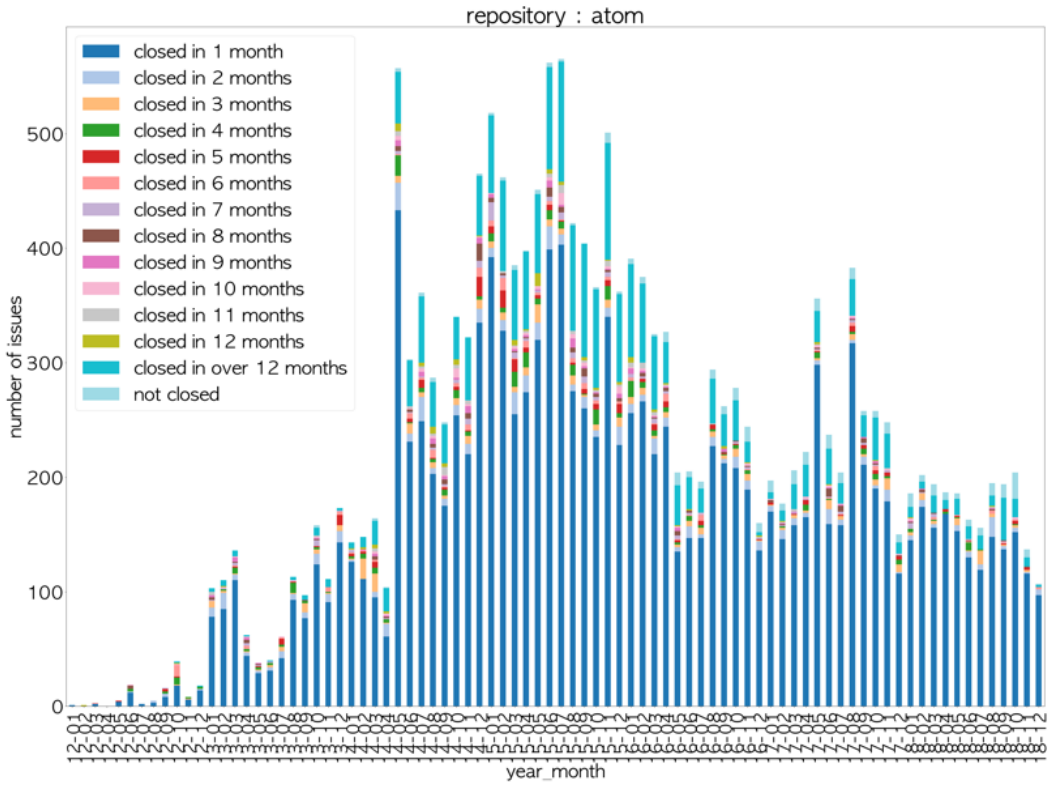

Figure 5-2(a-2). Patterns of transition in resolution status for each month by period. (a-2) Resolution period for the issues is extended; however, they are consistently closed within 12 months.

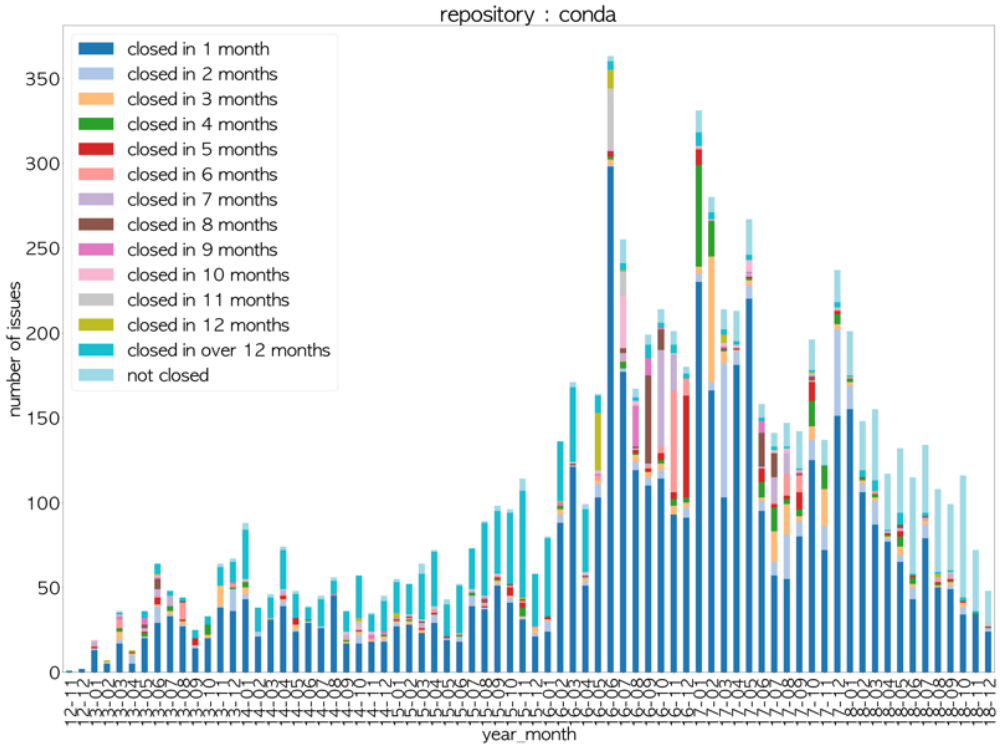

Figure 5-2(b). Patterns of transition in resolution status for each month by period. (b) Unresolved issues are present as new issues are created, thus extending from the middle stage to end of the relevant period.

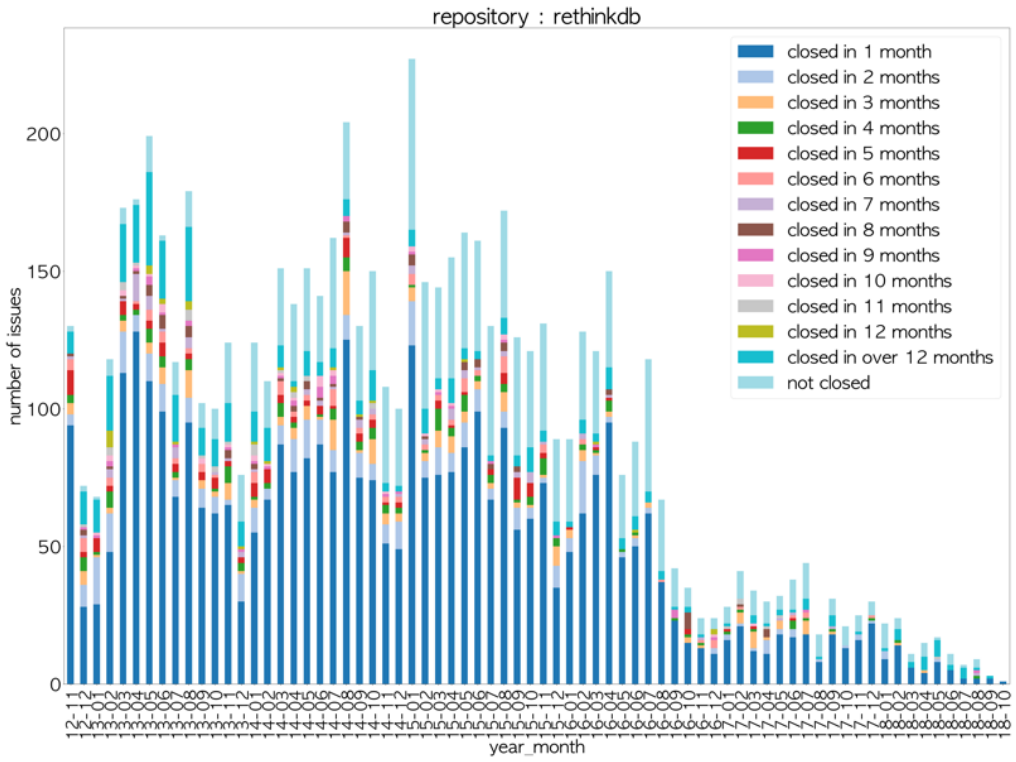

Figure 5-2(c-1). Patterns of transition in resolution status for each month by period. (c-1) Issue took considerable time to resolve. Unresolved issues from the middle stage are brought forward.

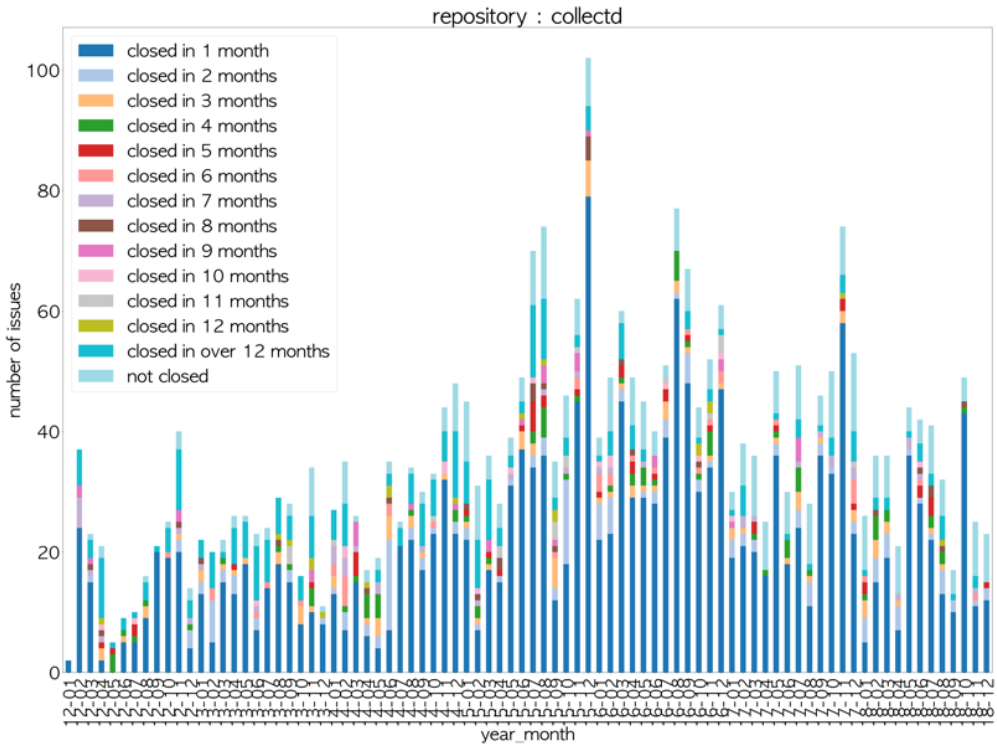

Figure 5-2(c-2). Patterns of transition in resolution status for each month by period. (c-2) Issues continuously created and take time to resolve leading to accumulation of unresolved issues.

The pattern in Figures 5-2(a-1) and 5-2(a-2) indicates a high resolution rate. In the case shown in Figure 5-2(a-1), the created issues are typically resolved in the same month they are created, while in the case shown in Figure 5-2(a-2), the issue resolution period is longer, but an issue is typically closed within 12 months. The pattern in Figure 5-2(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 5-2(c-1) and 5-2(c-2) show patterns wherein the resolution rate was slow from the beginning of OSS development. In particular, in the case shown in Figure 5-2(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.

5.4.2 Resolution Analysis Summary

Regarding 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.

5.4.3 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. Although there are various quality indicators, prompt and continuous issue resolution are two indispensable ones. Therefore, it is necessary to observe two operations, namely, the occurrence of and response to events.

Occurrences and responses that can be observed in the current OSS include the 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. Although 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 the prediction of quality by long- term observation. There are quite a few OSS that corporate business desire to adopt, regardless of their 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 5-2 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.

Table 5-2. 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

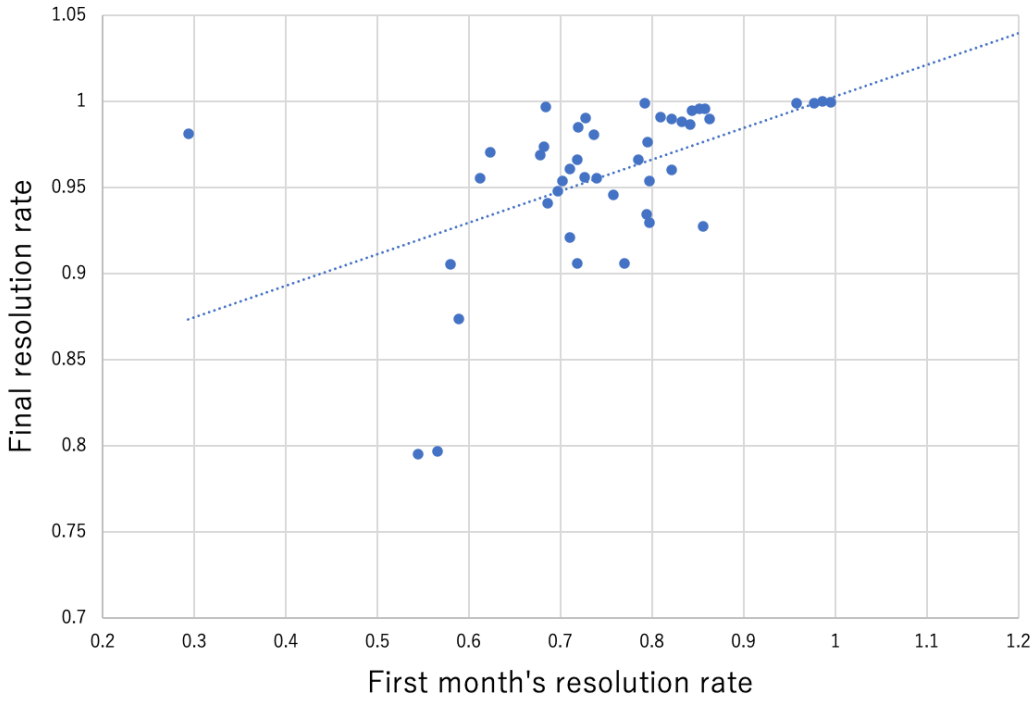

Figure 5-3. Correlation between final resolution rate and that of the first month after issue creation

It is clear from the results in Table 5-2 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 5-3 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.

5.5. Chapter Conclusion

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 would 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 the 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 the 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.

As a result, the resolution rate of the OSS project's issue response and the status of promptness and continuity are analyzed, and the resolution rate of the issues at the early stage (first month) regarding quality including support capability in actual use. There is a high correlation with the final solution rate, which leads to the fact that it can be used as knowledge for introduction decisions.

6. Conclusion

The purpose of this study was to identify the issues of OSS adoption decision-making and propose the technology to support the judgment, considering the situation that the use of OSS is indispensable in the system development of the company. In this study, we organized the concept of the vague behavior of the actual workplace and proposed a framework for adoption decision making from an OSS quality perspective, and we analyzed quantitatively the actual activities of the OSS development community to extract the knowledge.

Based on the issues of related research, the following three questions were articulated as the research themes.

- Research question 1: “Isn't it possible to clarify the decision-making procedure by extracting the axes and factors of OSS adoption evaluation and by creating a structured map to overview the OSS to be adopted?”
- Research question 2: “By looking at the status of the issue session of the OSS development projects by the OSS developer community, is it possible to extract an index that can quickly determine whether OSS can be used before starting a detailed examination?”
- Research question 3: “By looking at the resolution rate and the response promptness and response continuity of the issue session of the OSS projects by the OSS developer community, is it possible to extract knowledge to predict the final quality including the support capability in actual use?”

In the first study, the significance was to organize a case-by-case process in

decision-making for OSS adoption of business projects in the actual workplace.

As a result, a method for organizing the evaluation axes and factors of OSS adoption evaluation and the structure of the evaluator hierarchy was studied. A structured map was proposed that defines the positioning of the OSS evaluated from the perspective of technology, products, business project, company operation, and intellectual property creation and application, so that the project managers responsible for the entire system development can get an overview of the adoption evaluation.

In the practical consequences of the company, the adoption is decided based on the importance priority of the desired OSS, for example, the function, quality, and intellectual property matters; however, if the positioning is the same, the OSS with better quality will be adopted. Therefore, we decided to dig deeper into what and how OSS software quality, which is a key factor in hiring decisions, should be evaluated. Here, from the viewpoint of OSS usage, we defined the quality of OSS as "the resolution rate of issues processed by OSS developers as well as the promptness and continuity of doing so."

In the second study, the significance was the proposal of a framework for adoption decision making from an OSS quality perspective. Software quality indicators, which are a key factor in adopting OSS, were explored in depth.

As a result, from the analysis of the issue sessions of the OSS development projects, a 9-quadrant map named T-model was defined, focusing on the trends expressed by curve shape and the divergence time, in terms of the cumulative number of issues raised and resolved. Then, the mapped OSSs in the T-model were examined from the perspective of the final resolution rate. The axes to be related to software quality are the activeness of the developer community and its maturity of the technological innovation. It is proposed to

utilize the T-model as an index to quickly judge whether the target OSS can be used before starting a detailed examination.

Our proposed quality indicator model, the T-model, has shown the possibility of speedy decision-making regarding whether to adopt the desired OSS for software development of corporate information system. This T-model simply indicates “a region where there is generally no quality problem.” It does not indicate that the OSS project leaks from this region are “unavailable” for the target system development. Regarding the OSS project revealed from this T-model region, it is only necessary to assess the availability as it has been presently, by comparing with other various indexes, such as the degree of matching with the specifications in system development, required quality, required delivery time, and skills of the development team.

In the third study, the significance was to quantitatively analyze the actual activities of the development community and extract knowledge from the perspective of digging deeper into the axis of the activeness of the OSS development community. The indicators to predict OSS software quality, including support capability in actual use, which are a key factor in adopting OSS, should be explored in depth.

As a result, the resolution rate of the OSS project’s issue response and the status of promptness and continuity were analyzed, and the resolution rate of the issue at the early stage (first month) regarding quality including support capability in actual use. There is a high correlation with the final solution rate, which leads to the fact that it can be used as knowledge for introduction decisions.

Future study themes are as follows:

- The evaluation factors and axes of OSS adoption are influenced by the

progress of the open innovation strategy and tactics of the projects and the company. The factors for each evaluation layer and axis will be examined continuously.

- Regarding the OSS project revealed from this T-model region, it is necessary to analyze the availability with additional investigation. The method could be the way as it has been presently. The additional indicator could be extracted as the ratio of the pro-deviation and post-deviation periods, and the difference between the increase slope of created and closed issues. These will be further studied.
- 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, 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.

Moreover, another future study theme is the quantitative analysis from the remaining one axis, "the maturity of the technological innovation," out of the two axes of indexes for OSS software quality.

In this dissertation, a method to define the positioning of the target OSS in the OSS adoption evaluation is proposed. And a model consisting of two axes, which are the activeness of the developer community and its maturity of the technological innovation, is proposed to quickly judge from software quality of whether the target OSS can be used before starting a detailed examination. The knowledge to judge better quality OSS including support capability in actual use from the perspective of the OSS usage side is extracted. By utilizing this method and model and knowledge in the decision-making

process for adopting of OSS, we believe that it will contribute to the improvement of the productivity and efficiency of system development in business projects.

Acknowledgements

I would like to express my deepest appreciation to Prof. Tsuda whose comments and suggestions were of inestimable value for my dissertation. Throughout my time at Tsukuba University, he has been supporting me with great enthusiasm and patience. I am also deeply grateful to Prof. Tatsumoto and Prof. Kino for their thoughtful and practical comments and suggestions on this dissertation. My gratitude extends to my dissertation committee members, Prof. Yoshida, Prof. Morimoto, for their insightful comments and questions to my dissertation.

I would also like to thank the current and former Tsuda laboratory members for gratefully acknowledge their works, especially Dr. Masuda and Dr. Shida, also Prof. Fujita and Prof. Kato. Without their guidance and persistent help, this dissertation would not have been possible. I would like to offer my special thanks to Prof. Takahashi, and Prof. Suzuki. Besides, without the encouragement from Dr. Yanagihori, Dr. Sekiguchi, Dr. Ikoma, Dr. Murakami, Dr. Tanaka, and Dr. Tsujii, this dissertation would not have materialized.

Finally, I would like to thank my family for their love, support, and warm encouragement. In particular, I appreciate my wife Emiko and my daughter Shiho for their support, understanding, and encouragement throughout my long PhD journey. Thank you very much.

References

[Abdullah et al. 2009]

Abdullah R et al. The challenges of open source software development with collaborative environment. International Conference on Computer Technology and Development (ICCTD). Kota Kinabalu; 13-15 Nov. 2009. p. 251 - 255 ,2009

[Aberdour 2007]

Mark Aberdour. Achieving Quality in Open Source Software. IEEE.Jan-Feb 2007. p. 58-64 ,2007

[Akatsu 2012]

Akatsu S., Research on intellectual property management in open innovation development project, Department of Intellectual Property Strategy, Tokyo University of Science graduate school Intellectual Property Project Research papers, 2011, in Japanese

[Akatsu et al. 2018]

Akatsu S, Fujita Y, Kato T, Tsuda K. Structured analysis of the evaluation process for adopting open-source software. Procedia Comput. Sci. 2018 Jan; 26:1578-86, 2018

[Akatsu et al. 2020]

Akatsu S, Masuda A, Shida T, Tsuda K. A Study of Quality Indicator Model of Large-Scale Open Source Software Projects for Adoption Decision-Making, Procedia Comput. Sci. 2020 176:3665-72, 2020

[Akatsu et al.-2 2020]

Akatsu S, Masuda A, Shida T, Tsuda K., A Study of Quality Prediction for Large-Scale Open Source Software Projects, Artificial Intelligence Research, 2021, Vol. 10, No.1:33-41, 2020

[Android 2020]

Android [Internet] [cited 2020 Mar 26] Available from:

https://www.android.com/.

[Apache 2020]

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

[Apache-License 2021]

Apache Licenses [Internet]. [cited 2021 April 23] Available from: https://www.apache.org/licenses/

[Atieh and Riza 2011]

Atieh K and Riza S. The process of quality assurance under open source software development. IEEE Symposium on Computers & Informatics. 20-23 March 2011. p. 548-552

[Badashian and Stroulia 2016]

Ali Sajedi Badashian and Eleni Stroulia. Measuring user influence in GitHub. Proceedings of the 3rd International Workshop on Crowd-Sourcing in Software Engineering - CSI-SE '16, pp. 15–21, 2016

[Bahamdain 2015]

Bahamdain SS. Open Source Software (OSS) Quality Assurance: A Survey Paper, Procedia Computer Science, 2015 – Elsevier, 2015

[Bitbucket 2020]

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

[Blincoe et al. 2016]

Kelly Blincoe, Jyoti Sheoran, Sean Goggins, Eva Petakovic, and Daniela Damian. Understanding the popular users: Following, affiliation influence and leadership on GitHub. Information and Software Technology, Vol. 70, pp. 30–39, 2016

[Boehm 1981]

B.W. Boehm, Software engineering economics, Prentice-hall Englewood Cliffs. NJ, Vol.197, 1981

[BSD-2 2021]

The 2-Clause BSD License [Internet]. [cited 2021 April 23] Available from: https://opensource.org/licenses/BSD-2-Clause

[BSD-3 2021]

The 3-Clause BSD License [Internet]. [cited 2021 April 23] Available from: https://opensource.org/licenses/BSD-3-Clause

[CabinetSecretariat 2007]

Cabinet Secretariat Intellectual Property Strategy Headquarters, Promotion Measures of Intellectual Creation Cycle, Intellectual Property Strategy Headquarters, 2007, in Japanese

[Charette 2009]

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

[Chesbrough 2003]

Chesbrough H. Open Innovation: The New Imperative for Creating and Profiting from Technology, Harvard Business Review Press, 2003

[Chesbrough 2006]

Chesbrough H., Open Business Models: How To Thrive In The New Innovation Landscape, Harvard Business Review Press, 2006

[Chesbrough 2010]

Chesbrough H. Open Services Innovation: Rethinking Your Business to Grow and Compete in a New Era, Jossey-Bass, 2010

[Chesbrough and Appleyard 2007]

Henry Chesbrough and Melissa Appleyard, Open innovation and strategy, California Management Review, 2007

[Cubranic and Booth 1999]

Cubranic D and Booth K.S. Coordinating open-source software development. IEEE 8th International Workshops on Enabling Technologies, 1999

[Dabblish et al. 2012]

Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. Social coding in github: transparency and collaboration in an open software repository. In Proceedings of the ACM 2012 conference on computer supported cooperative work, pp. 1277–1286. ACM, 2012

[Dewan et al. 2004]

Dewan, Rajiv M and Freimer, Marshall L and Seidmann, Abraham and Zhang, Jie, Web portals: Evidence and analysis of media concentration, Journal of Management Information Systems. Taylor & Francis, Vol.21, No. 2, pp. 181–199, 2004

[Dindin et al. 2007]

Dindin W, Alexander S, Dietmar W and Stefan B. Aspects of software quality assurance in open source software projects: two case studies from Apache project. 33rd EUROMICRO Conference on Software Engineering and Advanced Applications. Lubeck; 28-31 Aug 2007. p. 229 - 236., 2007

[Franch et al. 2013]

Franch, X. [et al.]. Managing risk in open source software adoption. A: International Joint Conference on Software Technologies. "Proceedings of the 8th International Joint Conference on Software Technologies: Reykjavík, Iceland, 29-31 July, 2013". Scitepress, 2013, p. 258-264., 2013

[Gacek and Arief 2004]

Gacek C and Arief B. The many meanings of open source. IEEE. Jan-Feb 2004. p. 34-40 ,2004

[Gastwirth 1972]

Gastwirth, J. L., The Estimation of the Lorenz Curve and Gini Index, The Review of Economics and Statistics. The MIT Press, Vol.54, No. 3, pp. 306–316, August, 1972

[Gerner et al. 2005]

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

[Git 2020]

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

[GitHub 2020a]

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

[GitHub 2020b]

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

[GitHubAPI 2020]

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

[Glynn et al. 2005]

E. Glynn; B. Fitzgerald; C. Exton, Commercial adoption of open source software: an empirical study, 2005 International Symposium on Empirical Software Engineering, 2005.

[GPL 2021]

GNU General Public License [Internet]. [cited 2021 April 23] Available from: https://www.gnu.org/licenses/gpl-3.0.en.html

[Guimara et al. 2013]

A. LS. Guimara˜es, H. J. Korn, N. Shin, and A. B. Eisner, The life cycle of open source software development communities, Journal of Electronic Commerce Research, Vol. 14, No. 2, pp. 167, 2013

[Heisis 2007]

J. J. Heiss. The meanings and motivations of open-source communities. Aug 2007, from Oracle, 2007

[Higashi 2009]

Toshikazu Higashi, What is Koto Marketing? – New Perspective on Customers -, Ryutsukagaku Daigaku Ronbunshu Ryustu Unei Hen, Dai21kan2gou, 2009, 115-127, 2009, in Japanese

[IPA 2005]

IPA Information-technology Promotion Agency Japan, Investigation of legal risks of open source software in business use, IPA Information-technology Promotion Agency Japan, 2005, in Japanese

[IPA 2013]

IPA Information-technology Promotion Agency Japan, Common Frame 2013 -Realization of a "usable" system that works with management and business departments, IPA Information-technology Promotion Agency Japan, 2013 in Japanese

[IIBA 2015]

IIBA International Institute of Business Analysis, BABOK: A Guide to the Business Analysis Body of Knowledge, IIBA International Institute of Business Analysis, 2015

[Izquierdo et al. 2015]

Javier C´anovas Izquierdo, Valerio Cosentino, and Jordi Cabot. Attracting contributions to your github project. 2015. [cited 2020 Mar 26] Available from: https://www.researchgate.net/publication/291172663_Attracting_Contributions_to_your_GitHub_Project

[JapanPatentAttorneysAssocitation 2006]

Japan Patent Attorneys Association, 2nd Committee of the 2005 Software Committee. , Open Source Software License and Patent Rights, Monthly "Patent" Vol. 59 No. 6, 2006, in Japanese

[Jensen 2007]

C. Jensen, W. Scacchi, Role migration and advancement processes in OSSD projects: A comparative case study, Proceedings of the 29th

international conference on Software Engineering, IEEE Computer Society. pp. 364–374, 2007

[Kitayama 2009]

Kitayama S., Social Network Analysis of Communities in Organization, The Journal of Communication Studies. Japan, 29, 3-16, 2009. in Japanese

[Kobayakawa 2020]

Kobayakawa N, A Study on Acquiring Contributors in Open Source Software, Doctoral Dissertation Tsukuba University, 2020

[Kobayakawa and Yoshida 2017]

Naoki Kobayakawa and Kenichi Yoshida. How github contributing. md contributes to contributors. In 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), Vol. 1, pp. 694–696. IEEE, 2017

[Kobayakawa and Yoshida 2019]

Naoki Kobayakawa and Kenichi Yoshida. Study on influencers of cryptocurrency follow-network on github. In Pacific Rim Knowledge Acquisition Workshop, pp. 173–183. Springer, 2019

[Kobayakawa et al. 2020]

Naoki Kobayakawa, Mitsuyoshi Imamura, Kei Nakagawa, and Kenichi Yoshida. Impact of cryptocurrency market capitalization on open source software participation. Journal of Information Processing, Vol. 28, pp. 650–657, 2020

[Krogh and Hippel 2006]

G Von Krogh, E Von Hippel, The promise of research on open source software, Management science, 2006

[Krogh et al. 2012]

Georg Von Krogh, Stefan Haefliger, Sebastian Spaeth, and Martin W Wallin. Theory and Review Carrots and Rainbows : Motivation and Social Practice in Open Source Software Development. Vol. 36, No. 2,

pp. 649–676, 2012

[Linux 2020]

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

[LinuxFoundation 2011]

Linux Foundation, Overview of Open Source Compliance End-to-end Process, Linux Foundation, 2011

[Masuda 2019]

Masuda A., A Study on Revitalizing Software Development Team, Doctoral Dissertation Tsukuba University, 2019

[Masuda et al. 2017]

Ayako Masuda, Tohru Matsuodani, and Kazuhiko Tsuda., A Comparative Study Using Discriminant Analysis on a Questionnaire Survey Regarding Project Managers 'Cognition and Team Characteristics., IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), Vol.2, pp.643–648, 2017

[Masuda et al. 2018]

A. Masuda, C. Morimoto, T. Matsuodani, and K. Tsuda, A Study of Measurement for Development Efficiency in Large Scale Open Source Software Projects, IEEJ Transactions on Electronics, Information and Systems. Japan, Vol.138, No.8, pp. 1011—1019, August 2018

[Masuda et al. 2019]

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, 2019

[Munaiah et al. 2017]

Munaiah Nuthan, Kroh Steven, Cabrey Craig and Nagappan Meiyappan, Curating GitHub for engineered software projects, Empirical Software Engineering, Springer, Vol.22, No. 6, pp. 3219–3253, 2017

[MySQL 2020]

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

[Nakamura 2019]

K. Nakamura, Index for measuring the Income Gap: the Gini coefficient and the Lorenz curve, http://www.pref.toyama.jp/sections/1015/ecm/ back/2005apr/shihyo/ (accessed: Jan 7, 2019)

[National Tax Agency 2020]

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

[Nobeoka 2006]

Nobeoka K., Introduction to MOT (Technology Management), Nihon Keizai Shimbun, 2006, in Japanese

[OSI 2013]

OSI (Open Source Initiative), available from http://opensource.org/; Internet; Accessed 20 September 2013

[Otte et al. 2008]

Otte T, Moreton R and Knoell D. Applied quality assurance methods under the open source development model. Computer Software and Applications IEEE. Turku; July 28 -Aug 1, 2008. p. 1247 – 1252, 2008

[Pekka et al. 2009]

Pekka A, Janne M, and Eila O. Model-driven open source software development - the open models approach. International Conference on Software Engineering Advances (ICSEA). Portugal; 20-25 Sep

2009. p.185-190 ,2009

[Perl 2020]

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

[PHP 2020]

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

[Python 2020]

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

[Radlinski 2011]

Radlinski L. A conceptual Bayesian net model for integrated software quality prediction, Annales UMCS Informatica AI XI, 4 (2011) 49–60

[Stefano et al. 2005]

Stefano C, Fabio M and Maria P. From planning to mature: on the determinants of open source take-off. July 2005, Marco Fanno G. G. Schulmeyer & J. I. McManus, Handbook of software quality assurance 4th edition, 2005

[Tatsumoto 2021]

Tatsumoto H. Platform Strategy for Global Markets: Strategic Use of Open Standards and Management of Business Ecosystems, Springer, 2021

[Thung et al. 2013]

Ferdian Thung, Tegawende F. Bissyand´e, David Lo, and Lingxiao Jiang. Network structure of social coding in GitHub. Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR, pp. 323–326, 2013

[Tofu 2011]

Tofu Degawa, Perfect MOT, Shuwa-system, 2011, in Japanese

[Tosun et al. 2010]

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), 2010

[Ui 1995]

Tetsuo Ui, "Decision Support and Groupware"　Kyoritsu Publishing, 1995, in Japanese

[Vargo and Lusch 2008]

Stephen L. Vargo & Robert F. Lusch, Service-dominant logic: continuing the evolution, Journal of the Academy of Marketing Science, March 2008, 36:1–10, 2008

[Ware 2002]

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

[Xing et al. 2005]

Fei Xing, Ping Guo, M.R.Lyu, A Novel Method for Early Software Quality Prediction Based on Support Vector Machine, Proceedings of the 16th IEEE International Symposium on Software Reliability Engineering (ISSRE 2005), 2005

[Xiong et al. 2009]

Xiong C.J et al. A model of open source software maintenance activities. IEEE International Conference on Industrial Engineering and Engineering Management. Hong Kong; 8-11 Dec 2009. p.267-271, 2009

[Xu et al. 2005]

J. Xu, Y. Gao, S. Christley, and G. Madey, A topological analysis of the open source software development community, System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on, IEEE. pp. 198a–198a, 2005

[Yoshitaka et al. 2017]

Yoshitaka, Kuwata. Toru, Ishizuka. Shigetoshi, Yokoyama. Kento, Aida.; 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), 2017

[Zhou & Mockus 2012]

M. Zhou, A. Mockus, What make long term contributors: Willingness and opportunity in OSS community, Proceedings of the 34th International Conference on Software Engineering, IEEE Press. pp. 518–528, 2012

Related Achievement List

Akatsu S, Fujita Y, Kato T, Tsuda K., Structured analysis of the evaluation process for adopting open-source software, Procedia Computer Science, 2018 126:1578-1586

Akatsu S, Masuda A, Shida T, Tsuda K., A Study of Quality Indicator Model of Large-Scale Open Source Software Projects for Adoption Decision-Making, Procedia Computer Science, 2020 176:3665-3672

Akatsu S, Masuda A, Shida T, Tsuda K., A Study of Quality Prediction for Large-Scale Open Source Software Projects, Artificial Intelligence Research, 2021, Vol. 10, No.1:33-41

Akatsu S, Fujita Y, Tsuda K., A study on the structured analysis of the evaluation process for adopting open-source software, In Japanese, The Japan Society for Management Information, National Conference of JASMIN 2013 Autumn

Akatsu S, Fujita Y, Kato T, Tsuda K., A study on the structured evaluation criterion map for adopting open-source software in consideration of the management resources, In Japanese, The Japan Society for Management Information, National Conference of JASMIN 2014 Spring