Industry Questions About Open Source Software in Business: Research Directions and Potential Answers

Akinori Ihara*, Akito Monden*, Ken-ichi Matsumoto*
*Graduate School of Information Science, Nara Institute of Science and Technology, JAPAN
Email: (akinori-i, akito-m, matumoto)@is.naist.jp

***Abstract*—As open source software (OSS) has become an integral part of today's software businesses, many software companies rely on OSS to develop their customer solutions and products. On the other hand, they face various concerns in using OSS, such as technical support, quality, security and licensing issues. This paper focuses on OSS-related FAQ in industry, and tries to answer them or to provide research directions based on lessons learned from recent mining OSS repository researches.**

I. Introduction

Today's many software businesses rely on open source software (OSS) as it has now become an essential infrastructure in various computer environments. Typically, recent mobile phone companies are tackling Android operating system, which is now the most widely used smartphone platform in the world. Software/game companies and enthusiasts are also developing various applications for Android. As of July 2013, the Google play repository reached 1 million applications and 50 billion downloads [32].

While OSS enables companies to develop software systems at low cost, numbers of concerns are annoying the companies because using OSS is quite different from conventional software development. An industry survey of 916 Japanese software companies in 2009, conducted by the Information-technology Promotion Agency, Japan, has revealed numbers of questions (Table I) in using OSS [12], such as technical support, quality, security and licensing issues. These questions come from the nature of OSS — there are various major/minor versions and branches, with a huge number of security/bug patches available, while there is neither a person who is responsible of the quality of OSS, nor a customer service desk to get a technical support. This paper tries to answer industry questions based on lessons learned from mining software repositories (MSR) researches. Many OSS projects provide public software repositories — typically, source code repository, bug tracking repository and mailing list repository; and, this enables MSR researches to get useful information from the repositories for OSS developers and users.

II. OSS User Company Survey

This section introduces OSS user company survey conducted by Information-technology Promotion Agency, Japan [12]. Participants for the survey are 916 Japanese IT companies including software companies (556), IT service companies (121), Internet-related service companies (62) and unknown (177). 536 of them locate at metropolitan area, and 306 are at provincial area.

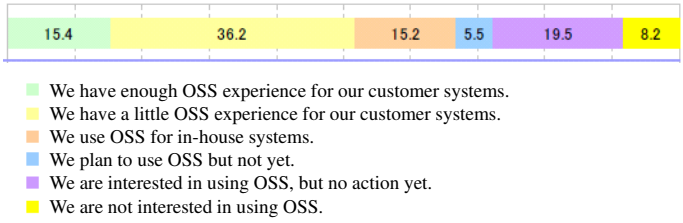

Fig. 1: Experience in using OSS [12]

Figure 1 shows experience in using OSS. Notably, 51.6% of the surveyed companies have experience in using OSS for their customer systems, and 66.8% including in-house systems. This indicates that many software companies rely on OSS to develop their customer solutions, and OSS are now an integral part of today's software businesses.

Table I shows major questions or concerns when using OSS in industry. The column "% Companies" indicates the percentage of companies who have posed each question. From next Section, we will seek for potential solutions for these questions from past MSR studies.

III. Potential Solutions for Industry Questions

A. Question 1: How can we get rapid technical help from an OSS community?

This question was posed by 67.3% of the surveyed companies [12]. Industry developers often face issues to be solved when they use OSS in their businesses. As there is no help desk for OSS in general, developers need to somehow get solutions from an OSS community.

In order to bridge the gap between developers and users, mailing lists have been used [10][25][23]. With growth of social Q&A sites such as the StackExchange network (e.g., StackOverflow), many people turn their interests to such sites. These sites are rapidly changing the way of collaborating between developers and users [30][2][4][22].

As a suggested answer or a research direction, recent MSR researches have focused on online software forums to get relevant solutions for a particular issue [9]. It is often the case that an issue faced by one developer or user has been faced by many others before. Therefore, one may find solutions from thread discussions in an online forum. To find a relevant solution quickly, Gottipati et. al. proposed a semantic search engine framework to process software threads and recover relevant answers according to user queries [9].

TABLE I: Six questions or concerns when using OSS in industry.

Questions	% Companies
Question 1: How can we get rapid technical help from an OSS community?	67.3%
Question 2: How much longer does an OSS project sustain?	58.8%
Question 3: How can we fix bugs and/or add new functions to OSS?	43.4%
Question 4: How can we understand and identify OSS licenses?	34.8%
Question 5: Which product and which version should we use?	32.8%
Question 6: How can we assess the quality of OSS products and the maturity of OSS projects?	29.5%

B. Question 2: How much longer does an OSS project sustain?

This question was posed by 58.8% of the companies [12]. It is often the case that an OSS project suddenly disappears or no longer continues updating products. For example, a famous web browser Netscape was discontinued and all support was terminated on March 2008 due to losing market share. As another example, the GIMP project, which started as an academic project, has stopped, because the creators left the university for work, and mostly ended their relationship with the GIMP. After all, the project was almost stopped for more than a year until someone else took over its control. This indicates that the coordination activity by core-developers is important to keep attracting people [21][35].

As a suggested research direction, activity metrics of developer/user communities [33] could be used to assess the project sustainability. If activity became low, it can be a symptom of project decay. In addition, some MSR researches focus on communication healthiness in the developer and user communication network [15][24]. It turned out that communication between user and developer was not active enough in the end of Netscape project. On the other hand, successful community (such as Apache) had many coordinators, who act as a bridge between user and developer communities [15]. These results suggest that activities and communications in OSS user/developer communities should be measured to assess the project healthiness.

Another direction is to analyze OSS projects from two point of views "stickiness" and "magnetism." Yamashita et al. [34] found that some projects attract new developers (magnet), and some retain existing contributors (sticky). For example, the Homebrew is one of the successful projects having high stickiness and magnetism.

C. Question 3: How can we fix bugs and/or add new functions to OSS?

This question was posed by 43.4% of the companies. In case an industry developer finds a bug (failure) or wants to add a new functionality to an OSS product, debugging and adding functionality in source code is usually very difficult because the code has been developed by someone else in the OSS community. Therefore, a technique to assist enhancing OSS code is greatly demanded.

To answer this question, several techniques from MSR researches can be used. One technique is called bug localization, which identifies a particular location in source code where a fault is likely to be existing, using as input detailed description of failure occurrence (such as actions to reproduce the failure, the name of a function where the failure has occurred, and so on). To this end, several studies propose the use of Information Retrieval (IR) based classifiers to locate bugs [36][14][29][18]. For example, text mining techniques are used to manage keywords in the failure description and source code to identify fault location. Recently, various improvements to bug localization have been proposed, and now it has become a hot topic in MSR [19][31].

Another technique is called a co-change analysis or a logical coupling analysis, which identifies a set of source files that needs to be changed together using as input past source code commit logs [1]. This technique helps developers to fix a bug and/or adding functionality to source code. Co-change histories are also used in recent bug localization studies [28].

Feature location is also a useful technique for OSS users when they want to identify an initial location in the source code that corresponds to a specific functionality (feature) to be enhanced or fixed [7]. A user can input a natural language query, execution trace or source code artifact, to obtain a ranked or visualized output of source code fragments such as files, classes, methods, functions or statements.

D. Question 4: How can we understand and identify OSS licenses?

This question was posed by 34.8% of the companies. There is a wide range of variations in Open Source licenses; and, many users do not understand them correctly. As a reference model of OSS licenses, the Open Source Initiative (OSI)[1] provides the Open Source Definition (OSD) as follows. The OSI accepts about 70 OSS licenses and provides a categorized list of licenses. This list may help OSS users to understand the variety of licenses.

Another industry concern with OSS license is that a single OSS product often contains multiple licenses; and thus, identifying all licenses in the product is often very difficult. One solution to this concern is use of an OSS license matching system. German et al. [8] proposed a text mining approach to automatically identify all licenses in a given set of source files.

Inspection of industry software products for possible OSS license violations is also becoming increasingly important as more reusable OSS code becomes available online [17][11]. Currently, several services are available for OSS code detection and management such as Black Duck Software's Protex (www.blackducksoftware.com) and Palamida

[1]The Open Source Initiative, http://opensoource.org

TABLE II: Open Source Definition (OSD)

No.	Difinition
1.	Free Redistribution
2.	Source code
3.	Derived Works
4.	Integrity of the Author's Source Code
5.	No Discrimination against Persons or Groups
6.	No Discrimination against Fields of Endeavor
7.	Distribution of License
8.	License Must Not Be Specific to a Product
9.	License Must Not Restrict Other Software
10.	License Must Be Technology-Neutral

(www.palamida.com). Futhermore, the "provenance" of source code [6] is also an important issue to understand where each program fragment come from and where it goes.

E. Question 5: Which product and which version should we use?

This question was posed by 32.8% of the companies. There are a huge number of OSS products in the world, and there may be too many similar OSS products in each application domain. Therefore, it is not easy to find the most relevant OSS product for a particular business objective. In addition, many OSS products have multiple versions where reliability, functionality and compatibility are different. The latest version is often not a good choice since it may be not stable enough. Finding appropriate version is also a difficult problem for industry developers.

One answer to find a relevant product is to use software search engines, such as SPARS [2], Koders [3], Jarhoo [4]. These search engines are especially useful for searching for a particular software component. Another answer is to use an automatic software categorization system, which helps users to find similar OSS products [13].

Regarding finding appropriate version, we could learn from "the wisdom of the crowd." Mileva et al. [16] analyzed the frequency of use of 450 different library versions and developed a library recommender system called AKTARI [5]. They found that developers often downgrade to previous library because of bugs found in new version. Generally, if a specific version is used by many developers, we could consider it is of high quality.

Similarly, Sunada also followed the idea of "the wisdom of crowds," to clarify frequently-used libraries in Java open source software development [27]. He analized frequently-used libraries in many domains, and clarified domain-specific libraries, as well as domain-independent libraries such as basic function libraries (e.g. commons-collections, commons-lang) and logging libraries (e.g. commons-logging and log4j). Figure 2 shows the result for Enterprize applications in Business/Office domain.

[2]SPARS Project, http://sel.ist.osaka-u.ac.jp/SPARS/
[3]Koders.com, http://www.koders.com/
[4]Jarhoo, http://www.jarhoo.com/
[5]AKTARI, http://www.st.cs.uni-saarland.de/softevo/aktari.php

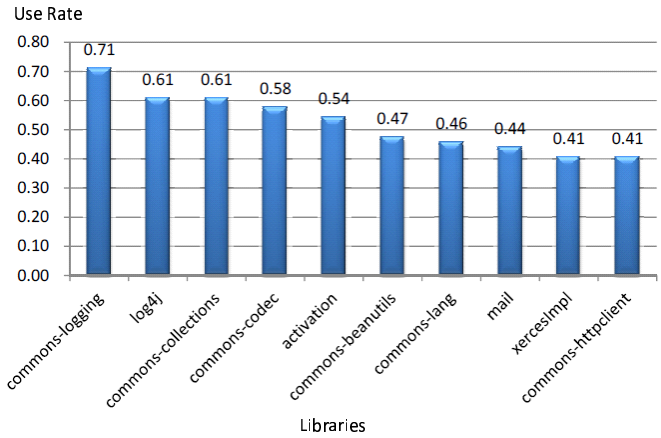

Fig. 2: Commonly used OSS libraries in Business/Office domain [27]

F. Question 6: How can we assess the quality of OSS products and the maturity of OSS projects?

This question was posed by 29.5% of the companies. When a company considers using an OSS product for their business, the company needs to assess the quality of the product as well as the maturity of the project. Also, the company often needs to explain the quality of the OSS product to their customers. To answer this question, we introduce QualiPSo (Quality Platform for OSS) project. In 2006, QualiPSo was launched to establish the reputation of reliability and quality for OSS by EC (European Communities). 21 organizations in 11 countries collaborated to investigate 7 fields (Quality/ Interoperability/ Factory/ Organization/ Life Cycle Matunity/ Business/ Legal) of OSS. MOSST (Model for OSS trust worthiness) and OMM (Open Source Maturity Model) were developed based on the result of MSR techniques and interview to industry developers who use OSS.

The MOSST is a customizable model for estimating the trust that OSS developers and end-users can have in the qualities of OSS products. One of the functions is to visualize the reliability of OSS. They assessed the reliability of the OSS based on over 90 metrics measured from source codes and binary code of OSS. Then it shows it in 3 levels (good, acceptable, poor). The MOSST provides advices to improve the reliability of OSS for OSS developers. Also, it helps the reputation of reliability and quality for OSS users. They have already applied in over 100 OSS projects to evaluate the MOSST as 14 March, 2012.

The OMM is a CMMI-like process model for OSS development. This process model aims to help in building trust in development processes of companies using or producing OSS. The OMM comprises trustworthy elements required for OSS development based on surveys and best practices from CMMI. The trustworthy elements are grouped into 3 maturities levels (basic, intermediate and advanced). Now, they are trying to establish the standard reliability evaluation index for OSS.

IV. DISCUSSION

リポジトリマイニングの研究では，OSS のメカニズムに関する分析やそのモデル化の研究が多く，ツール開発，及び，そのサービス化まで進んだ研究が少ない．サービス化まで進んでいる研究の一つとして，XX らは XX のサービス化を行っている．OSS を商用ソフトウェアに利活用する企業が XX サービスを利用することで，従来困難であった OSS の第三者評価が可能になる．このように，OSS が利活用される事例が増加していることから，MSR 研究者は利用者が課題を解決するためのユーザ指向研究に取り組む必要がある．

一方で，利用者も OSS 開発者，及び，支援ツールに頼るばかりではなく，OSS プロジェクトの生態を理解する努力が必要である．いくつかの企業では OSS をより深く理解するために，社員がプロジェクトに参加し，企業にフィードバックしている．しかし，OSS プロジェクトへの参加はハードルが高い．Steinmacher ら [26] は OSS プロジェクトへの参加に対する障害として複雑なコードの理解，ドキュメントの情報欠落，メンターの不足などが挙げられている．複雑なコード理解には XX や XX のサービスが提案されている．また，ドキュメントの作成，保守には相当なコストが費やされており，Dagenais らがドキュメント作成の作成と保守に必要となるリソースを最適にするための技法に関する知見を調査している．しかし，未だドキュメントの保守に関する研究は不十分である．メンターの推薦に関する研究も現在進められているが，実用化されているわけではない [20][5]．今後，プロジェクトに容易に参加を促すアプローチをマイニングから見出し，利用者が OSS プロジェクトと密接に付き合っていく方法を確立していくことが必要である．

V. CONCLUSION

This paper introduces commonly-asked industry questions about OSS for a business use, and tries to answer them or to provide research directions based on lessons learned from recent MSR researches. We hope that industry developers find their own answers in their context based on suggested answers in the paper. We also hope, in future, more powerful solutions will come up from further MSR researches to assist OSS users in various application domains.

ACKNOWLEDGMENT

This work has been conducted as a part of ”Research Initiative on Advanced Software Engineering in 2013” supported by Software Reliability Enhancement Center (SEC), Information Technology Promotion Agency Japan (IPA). Also, part of this research was conducted under Japan Society for the Promotion of Science, Grant-in-Aid for Young Scientists (B) (25730045), and Scientific Research (C) (22500028).

REFERENCES

[1] Dirk Beyer, “Co-change visualization applied to PostgreSQL and ArgoUML,” Proceedings of the 3rd International Workshop on Mining Software Repositories (MSR’06), pp.165-166, 2006.

[2] Andrew Begel, Jan Bosch, and Margaret-Anne Storey, “Social Networking Meets Software Development: Perspectives from GitHub, MSDN, Stack Exchange, and TopCoder,” IEEE Software, Volume.30, Issue.1, pp.52-66, 2013.

[3] Gerardo Canfora, Massimiliano Di Penta, Rocco Oliveto, and Sebastiano Panichella, “Who is going to mentor newcomers in open source projects?,” In Proceedings of the International Symposium on the Foundations of Software Engineering (FSE’12), 1-11 pages. DOI=10.1145/2393596.2393647 http://doi.acm.org/10.1145/2393596.2393647

[4] Andrea Capiluppi, Alexander Serebrenik, and Leif Singer, “Assessing Technical Candidates on the Social Web,” IEEE Software, pp.45-51, 2012.

[5] Barthélémy Dagenais and Martin P. Robillard, “Creating and evolving developer documentation: understanding the decisions of open source contributors,” In Proceedings of the International Symposium on Foundations of Software Engineering (FSE’10), 2010.

[6] Julius Davies, Daniel M. German, Michael W. Godfrey, and Abram Hindle, “Software bertillonage: finding the provenance of an entity,” Proceedings of the 8th Working Conference on Mining Software Repositories (MSR2011), pp. 183-192, 2011.

[7] Bogdan Dit, Meghan Revelle, Malcom Gethers, and Denys Poshyvanyk, “Feature location in source code: a taxonomy and survey,” Journal of Software: Evolution and Process, Vol. 25, No. 1, pp. 53-95, 2013.

[8] Daniel M. German, Yuki Manabe, and Katsuro Inoue, “A Sentence-Matching Method for Automatic License Identification of Source Code Files,” Proceedings of the 25th International Conference on Automated Software Engineering (ASE’10), pp. 437-446, 2010.

[9] Swapna Gottipati, David Lo, and Jing Jiang. “Finding relevant answers in software forums,” Proceedings of the 26th International Conference on Automated Software Engineering (ASE’11), pp. 323-332, 2011.

[10] Anja Guzzi, Alberto Bacchelli, Michele Lanza, Martin Pinzger, Arie van Deursen, “Communication in open source software development mailing lists,” Proceedings of the 10th Working Conference on Mining Software Repositories (MSR’11), pp.277-286, 2011.

[11] Katsuro Inoue, Yusuke Sasaki, Pei Xia, and Yuki Manabe, “Where does this code come from and where does it go? - integrated code history tracker for open source systems -,” In Proceedings of the International Conference on Software Engineering (ICSE’12), pp.331-341, 2012.

[12] Information-technology Promotion Agency, “A survey report on open source software based businesses (year 2009 edition)”. 2010.

[13] Shinji Kawaguchi, Pankaj K. Garg, Makoto Matsushita, and Katsuro Inoue, “MUDABlue: An automatic categorization system for open source repositories,” Journal of Systems and Software, Vol.79, No.7, pp.939-953, 2006.

[14] Dongsun Kim, Yida Tao, Sunghun Kim, and Andreas Zeller, “Where Should We Fix This Bug? A Two-Phase Recommendation Model,” IEEE Transaction Software Engineering, Volume.39, Issue.11, pp.1597-1610, 2013.

[15] Shinsuke Matsumoto, Yasutaka Kamei, Masao Ohira, and Kenichi Matsumoto, “A comparison study on the coordination between developers and users in FOSS communities,” Proceedings of theSocio-Technical Congruence (STC’08), pp.1-9, 2008.

[16] Yana Momchilova Mielva, Valentin Dallmeier, Martin Burger, and Andreas Zeller, “Mining trends of library usage,” Proceedings of the International Workshop on Principles of Software Evolution (IWPSE’09), pp.57-62, 2009.

[17] Akito Monden, Satoshi Okahara, Yuki Manabe, and Kenichi Matsumoto, “Guity or not guilty: Using clone metrics to determine open source licensing violations,” IEEE Software, Vol. 28, No. 2, 2011.

[18] Anh T. Nguyen, Tung T. Nguyen, Jafar M. Al-Kofahi, Hung V. Nguyen, and Tien N. Nguyen, “A topic-based approach for narrowing the search space of buggy files from a bug report,” Proceedings of the International Conference on Automated Software Engineering (ASE’11), pp.263-272, 2011.

[19] Ripon K. Saha, Matthew Lease, Sarfraz Khurshid, Dewayne E. Perry, “Improving bug localization using structured information retrieval,”Proceedings of the 28th International Conference on Automated Software Engineering (ASE2013), pp.345-355, 2013.

[20] Igor Steinmacher, Igor Scaliante Wiese, and Marco Aurélio Gerosa, “Recommending mentors to software project newcomers,” In Proceedings of the International Workshop on Recommendation Systems for Software Engineering (RSSE’12), pp.63-67, 2012.

[21] Bianca Shibuya and Tetsuo Tamai, "Understanding the process of participating in open source communities," In Proceedings of the Emerging Trends in Free/Libre/Open Source Software Research and Development (FLOSS'09), pp.1-6, 2009.

[22] Leif Singer, Fernando Figueira Filho, Brendan Cleary, Christoph Treude, Margaret-Anne Storey, and Kurt Schneider, "Mutual assessment in the social programmer ecosystem: an empirical investigation of developer profile aggregators," Proceedings of the Conference on Computer supported cooperative work (CSCW'13), pp.103-116, 2013.

[23] Vandana Singh, Michael B. Twidale, and David M. Nichols, "Users of Open Source Software How do they get help?" Proceedings of the Hawaii International Conference on System Sciences (HICSS'09), pp.1-10, 2009.

[24] Param V. Singh, "The small-world effect: The influence of macro-level properties of developer collaboration networks on open-source project success," ACM Transaction on Software Engineering Methodology, Volume.20, Issue.2, Article 6, 2010.

[25] Sulayman K. Sowe, Ioannis Stamelos, and Lefteris Angelis, "Understanding knowledge sharing activities in free/open source software projects: An empirical study," Journal of Systems and Software, Volume.81, Issue.3, pp.431-446, 2008.

[26] Igor Steinmacher, Igor Scaliante Wiese, Tayana Conte, Marco Aurélio Gerosa, and David Redmiles, "The hard life of open source software project newcomers," In Proceedings of the International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE'14), pp.72-78, 2014.

[27] Takahiro Sunada, "Trends of library usage in different domains of Java software development," Master's Thesis, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-MT1151059, 2013 (in Japanese).

[28] Chakkrit Tantithamthavorn, Akinori Ihara, and Kenichi Matsumoto, "Using Co-change Histories to Improve Bug Localization Performance," Proceedings of 14th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD2013), pp. 543-548, 2013.

[29] Stephen W. Thomas, Meiyappan Nagappan, Dorothea Blostein, and Ahmed E. Hassan, "The Impact of Classifier Configuration and Classifier Combination on Bug Localization," IEEE Transaction Software Engineering, Volume.39, Issue.10, pp.1427-1443, 2013.

[30] Bogdan Vasilescu, Alexander Serebrenik, Prem Devanbu, and Vladimir Filkov, "How social Q&A sites are changing knowledge sharing in open source software communities," Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing (CSCW'14), pp.342-354, 2014.

[31] Shaohua Wang, Foutse Khomh, and Ying Zou, "Improving bug localization using correlations in crash reports," Proceedings of Working Conference on Mining Software Repositories (MSR'13), pp. 247-256, 2013.

[32] Christina Warren, "Google Play Hits 1 Million Apps," Mashable. Retrieved 4 June 2014.

[33] Hao Zhong, Ye Yang, and Jacky Keung, "Assessing the representativeness of open source projects in empirical software engineering studies," Proceedings of the 19th Asia-Pacific Software Engineering Conference (APSEC ' 12), pp. 808-817, 2012.

[34] Kazuhiro Yamashita, Shane McIntosh, Yasutaka Kamei, and Naoyasu Ubayashi, "Magnet or sticky? an OSS project-by-project typology," In Proceedings of the Working Conference on Mining Software Repositories (MSR'14). pp.344-347, 2014.

[35] Yunwen Ye, Kumiyo Nakakoji, Yasuhiro Yamamoto and Kouichi Kishida, "The Co-Evolution of Systems and Communities in Free and Open Source Software Development," Free/Open Source Software Development, IGI Publishing, Hershey,, pp.59-82. 2005.

[36] Jian Zhou, Hongyu Zhang, and David Lo, "Where should the bugs be fixed?," Proceedings of the 34th International Conference on Software Engineering (ICSE'12), pp.14-24, 2012.