OSMF: A Framework for OSS Process Measurement

Wikan Danar Sunindyo and Fajar Juang Ekaputra

Data and Software Engineering Research Group
School of Electrical Engineering and Informatics, Bandung Institute of Technology
Labtek V 2nd floor Ganesha Street 10 Bandung 40132 Indonesia
{wikan,fajar}@informatika.org

Abstract. An Open Source Software (OSS) project can be considered as a new type of business entity involving various roles and stakeholders, e.g., project managers, developers, and users, who apply individual methods. The project managers have the responsibility to manage the OSS development in a way that the OSS product can be delivered to the customers in time and with good quality. This responsibility is challenging, because the heterogeneity of the data collected and analyzed from different stakeholders leads to the complexity of efforts of the project managers to measure and manage OSS projects. In this paper, we propose a measurement framework (OSMF) to enable the project managers to collect and analyze process data from OSS projects efficiently. Initial results show that OSMF can help project managers to manage OSS business processes more efficient, hence improve the decision on OSS project quality.

Keywords: Open Source Software, process observation, project management, software quality.

1 Introduction

Both development and usage of Open Source Software (OSS) products are increasing exponentially and play a significant role in the software economy [4]. OSS development projects are business entities, which involve several different stakeholders, like project managers, developers, and customers [13].

Mockus *et al.* [8] describe four key characteristics of OSS projects: first, it consist of large number of volunteers [9]; second, anyone can choose tasks he/she wants to execute [3]; third, there is neither explicit system design nor explicit detailed design [5]; fourth, there is no project plan, schedule or list of deliverables [8].

Major OSS projects like FreeBSD divide project participants into three main roles: *core team members*, *committers* and *contributors* [14]. The Core Team is a small group of senior developers that act as project managers who are responsible for deciding about the overall goals and directions of the project. Committers are developers who have the authority to commit changes to the project repository. Contributors are people who want to contribute to the project, but do not have committer privileges.

Typically, project managers face challenges defining, manage and improve business process in OSS projects, since business processes in OSS projects are different than business processes in conventional software projects. However, business process

K. Mustofa et al. (Eds.): ICT-EurAsia 2013, LNCS 7804, pp. 71–80, 2013.

definition is still a key foundation to enable and improve business process observation by means of monitoring and controlling the status of OSS projects.

In this paper, we propose a measurement framework (OSS project measurement framework – OSMF) to manage (i.e., definition, observation, and control) business processes in OSS projects. The contribution of this framework is to guide the project manager in following the processes of engineering process observation.

For evaluating the OSMF, we use OSS project data from the Red Hat Enterprise Linux (RHEL) and Fedora projects as use cases. We evaluate the engineering processes in both projects, by checking conformance between the engineering process models from both projects to the designed process model. The results can show the frequent states of the process for project managers' decision.

We organize this paper as follows. After introduction, we explain related works that support the OSMF framework. The third section presents the research issues that we discuss in this paper. Section four shows the case we use in our work. Section five discusses about the results of our study. We summarize the results and lessons learned of our work in section six, and finally conclude our paper and present future work.

2 Related Work

This section summarizes related works on business process management and OSS.

2.1 Business Process Management

Business process management (BPM) is a management approach focusing on aligning all aspects of an organization with the requests and needs of clients. It is a holistic management approach [16] that promotes business effectiveness and efficiency while striving for innovation, flexibility, and integration with technology. An empirical study by Kohlbacher reveals that BPM helps organizations to gain higher customer satisfaction, product quality, delivery speed and time-to-market speed [6].

Reijers, van der Aalst and zur Muehlen used the notion of a life-cycle to distinguish between the various phases that a BPM initiative can go through [11, 15, 17].

This life-cycle distinguishes between six phases: namely analysis, design, implementation, enactment, monitoring, and evaluation phases. In the *analysis* phase, a set of requirements is developed for the business process in questions such as performance goals or intentions. In the *design* phase, the process activities, their order, the assignment of resources to activities and the organization structure are defined. In the *implementation* phase, the infrastructure for the business process is set up. The dedicated infrastructure is used to handle individual cases covered in the *enactment* phase. In the *monitoring* phase, counteractions are taken to deal with problematic situations depending on process metrics. The new requirements are taken as input in the next turn of the business process management life-cycle in the *evaluation* phase [10].

2.2 Process Mining

Process mining is a process management technique that allow for the analysis of business processes based on event logs. Process mining aims at improving this by providing techniques and tools for discovering process, control, data, organizational, and social structures from event logs[1].

In the area of software development project, Rubin et al. [12] proposed a process mining framework for software processes, which is based on fact that software development process are often not explicitly modeled and even chaotic. In this paper, we use this approach to develop our own framework for mining OSS processes.

3 Research Issues

The major challenges related to managing business processes in OSS projects are as follows, (1) better definition of OSS business processes, (2) more effective data collection for heterogeneous OSS business processes, (3) better analysis methods to give more precise results for project managers' decision making. From these challenges, we derive the research issues addressed in this paper as follows:

RI-1: How to describe the business aspects and their interactions in OSS projects. OSS project is a unique business entity, because it mixes the conventional business process and non-conventional business process.

We propose to describe business aspects in OSS projects and their interactions between business managers, project developers, and customers, so we get clearer view on the OSS projects business.

RI-2: How to evaluate measurement of the engineering processes data? The measurement of the OSS projects engineering process data can be done by collecting and analyzing engineering process data from OSS projects.

We propose to measure the bug history data by performing conformance check analyses between the actual process model and the designed process model. We defined two research hypotheses to be validated by the experiments as follows.

RH-1. RHEL and Fedora developers are following the naming and order of the bug states of the original Bugzilla life cycle. The changing of bug states from developers lead to a chain of bug status history that we can trace to find out the pattern of the bug status usually used by the developers in developing the OSS projects.

In this study, we generate process models from bug history data of two OSS projects, namely RHEL and Fedora and make conformance checking with the designed process model from Bugzilla. IF ϕ is the designed process model, and ψ is an OSS project, and P is a function to get the number of bug states from designed process model or from OSS projects, THEN we can formulate following null hypothesis.

$$\boldsymbol{H01}: \{\exists \psi \in (\text{RHEL, Fedora}) \mid P(\psi) = P(\phi)\} \quad \text{(eq. 1)}$$

[1] http://www.processmining.org

RH-2. OSS projects developers are using all bug states for each bug history in the same number of frequency. We want to measure and investigate the frequency of bug states used in one OSS project. IF s_i is a bug state and s_{i+1} is another bug state after s_i, and N is a function to obtain the frequency of bug states using in Fedora, THEN we can propose following null hypothesis.

$$\boldsymbol{H02}: \{\forall s_i,\ s_{i+1} \mid N(s_i) = N(s_{i+1})\} \quad \text{(eq. 2)}$$

4 Solution Approach

In this section we address the research issues in section 3.

4.1 Business Aspects of OSS Projects

Crowston et al. [2] suggested three aspects of the success of OSS projects, namely *developers' contribution*, the intensity of *software usage*, and the *quality of software products*. Based on this suggestion, we model the business aspects of OSS projects in three parts, namely *developer*, *customer*, and *business* parts. The causality model of the business aspects and their interactions is shown in Figure 2.

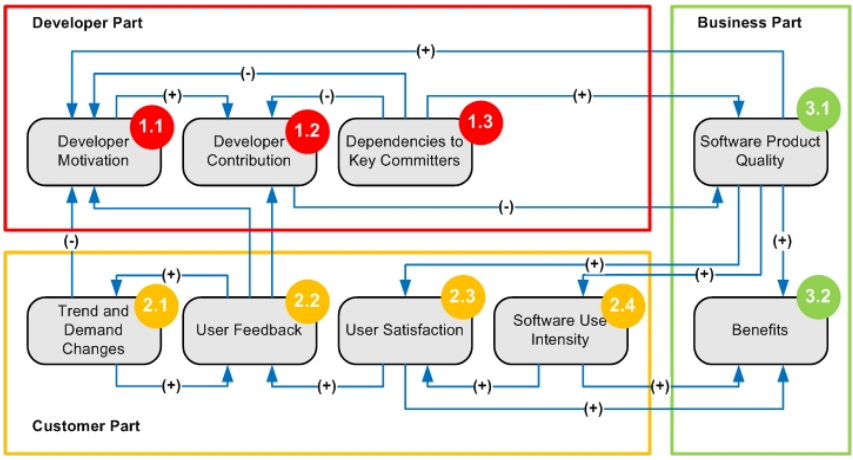

Fig. 1. Model of OSS Projects Business Aspects and Their Interactions

The developer part consists of three aspects. (1.1) *Developer motivation* is about the motivation of developers to join the developers' community to develop the product. (1.2) *Developer contribution* consists of contribution of developers to the OSS project, e.g., via source code management, developers' mailing list, or bug management systems. (1.3) *Dependencies to key committers* mean that the survivability of the OSS projects depends on very few active committers (key committers).

The customer part consists of four aspects. (2.1) *Trend and Demand Changes* from the customers can be submitted to the developers and be a part of software

requirements. (2.2) *User feedback* is a feedback from the users about functionality of the software that can be improved by the developers. (2.3) *User satisfaction* is a notification from the users that they satisfy with the functionality of the product. (2.4) *Software use intensity* is a intensity of software usage by the users.

The business part deals with software product quality and benefits of the software products. (3.1) a *software product quality* deals with the qualitative and quantitative measurement of product that comply with users' satisfaction. (3.2) the *benefits* of software product can be used as a selling point of the product.

4.2 OSMF Measurement Framework

To analyze and measure the OSS processes, we propose the OSMF measurement framework for supporting the OSS project managers with data collection and data analysis. Figure 3 shows the measurement framework for observing OSS processes, i.e. bug history data.

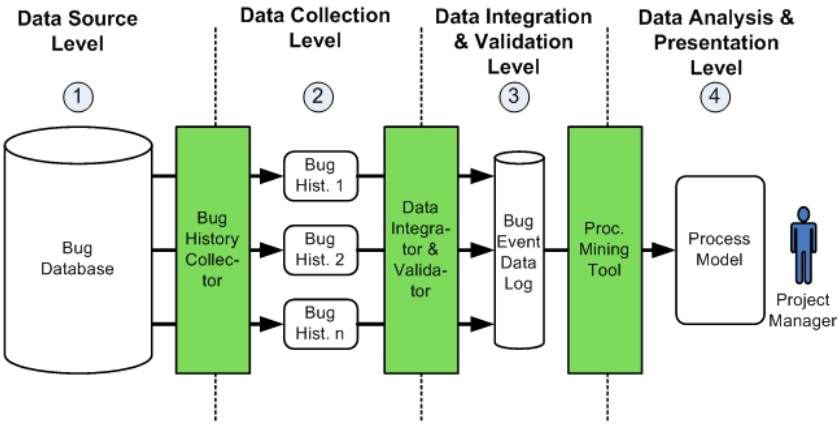

Fig. 2. OSMF Framework for OSS Processes Measurement

This framework consists of 4 levels, namely data source level, data collection level, data integration and validation level, and data analysis and presentation level.

(1) In the *data source level*, we have bug database which contains all bugs information that are used in software development.
(2) In *data collection level,* we extract and collect bug history data from bug history data using a bug history collector. The bug history collector is written in Java and use XML RPC[2] web service interface to access to the bug data.
(3) *Data integration and validation.* The collected bug history data is integrated and validated using data integrator and validator. The integration and validation process is implemented by using Engineering Knowledge Base (EKB) [1].

[2] `https://bugzilla.redhat.com/xmlrpc.cgi`

(4) *Data analysis and presentation*. The even data log from previous level is analyzed by using Process Mining tool. The results are presented to the project managers.

The study objects of this research are two different OSS projects, namely the Red Hat Enterprise Linux and Fedora projects. Both projects are developing Linux-based operating systems and under the same Red Hat project management.

Currently, in the Red Hat Bugzilla browser, there are in total 23.768 bugs reported RHEL version 6 (7.465 open bugs and 16.303 closed bugs) and 293.685 bugs reported Fedora (25.386 open bugs and 268.299 closed bugs).

4.3 Evaluation of Engineering Process Measurement Results

For analyzing the bug history data, we used a process analysis tool called ProM. This tool has capabilities to discover process model, make conformance checking between expected process model and the process model generated from actual data, and make performance analysis on process model for process improvement. There are a lot of plug-ins and algorithms to discover the process model from actual data. One of them is heuristics mining.

The heuristics mining is a process mining algorithm in ProM which is based on the frequency of the patterns. The most important characteristic of the heuristics mining is the robustness for noise and exceptions. We use the heuristics mining to analyze event log from bug history data to find out the process model from actual data, rather than designed process model.

We identified and addressed threats to internal and external validity of our evaluation results as follows.

Threats to Internal Validity. As we have conducted previous experiments using fewer data, there is a tendency of increasing of the numbers of states as new data is added. So we put more focus on the frequency of states taken during development, since the number of states can be unnecessary increasing, while the top states remain stable.

Threats to External Validity. In this study we focus on projects with similar characteristics that may raise concerns whether the results are also valid for other project contexts. While we assume our approach to hold for projects similar to our objects, further research work is necessary to investigate projects with strongly differing characteristics.

5 Results

In this section, we report the results of our study presented in section 4.

5.1 OSS Processes Measurement Framework

To observe software engineering processes from the bug database effectively, we applied the observation framework from figure 3. In the application, we take Bugzilla

report of RHEL and Fedora projects as data sources, Bug History Data Collector as an automated data collector, integrator and validator, and Process Mining (ProM) tool for data analysis and presentation.

We have implemented and used a Java-based Bug History Data Collector tool to collect, integrate, and validate bug history data from Bugzilla database. This tool can select bug ids based on the OSS projects and versions. As results, we have collected 1000 data sets from RHEL 6 and Fedora 14. These data will be analyzed for process model conformance checking with the Bugzilla life cycle.

5.2 Evaluation of Engineering Process Measurement Results

We analyze the number of states in the process models generated by ProM and count the frequency of each state for each RHEL version. We compare the results with the designed process model from Bugzilla life cycle and evaluate the actual data by answering the two hypotheses defined in section 3.

Table 1. Name and frequency of Bug States used in RHEL and Fedora 14, compared with designed process model from Bugzilla life cycle

States	Bugzilla LC	RHEL 6		Fedora 14	
		Occ. (abs)	Occ. (rel)	Occ. (abs)	Occ. (rel)
CLOSED	✓	**546**	**33.6 %**	**500**	**54.1 %**
ASSIGNED	✓	**255**	**15.7 %**	**208**	**22.5 %**
NEEDINFO	×	6	0.37 %	**73**	**7.9 %**
MODIFIED	×	**306**	**18.9 %**	**74**	**8.0 %**
REOPENED	✓	1	0.1 %	×	×
ON_QA	×	**259**	**16.0 %**	**43**	**4.7 %**
RELEASE_PENDING	×	×	×	1	0.1 %
NEW	✓	7	0.4 %	19	2.1 %
NEEDINFO_REPORTER	×	2	0.1 %	1	0.1 %
INVESTIGATE	×	2	0.1 %	×	×
VERIFIED	✓	**199**	**12.3 %**	2	0.2 %
FAILS_QA	×	×	×	1	0.1 %
ASSIGN_TO_PM	×	1	0.1 %	×	×
ON_DEV	×	8	0.5 %	2	0.2 %
POST	×	31	1.9 %	×	×
UNCONFIRMED	✓	×	×	×	×
RESOLVED	✓	×	×	×	×

RHEL and Fedora developers are following the naming and order of the bug states of the original Bugzilla life cycle. Table 1 shows the comparison of bug states used in the Bugzilla life cycle, RHEL 6, and Fedora 14. From Table 1 we can see different bug state names are used during addressing bug in different RHEL versions.

RHEL and Fedora developers are using all bugs states for each bug history in the same frequency. Table 1 shows the frequencies of the using of each bug state in the bug history. We can see that the frequencies for different bugs in one RHEL version are not similar. The usage of some states is more frequent then of other states.

6 Discussion

In this section, we discuss our results based on the research issues.

6.1 OSS Processes Measurement Framework

We followed the observation framework we defined earlier to improve the engineering process in OSS projects. The benefit of this framework is an effective and systematic approach to collect and analyze data from the bug database to support the project managers' decision making to improve the process quality in OSS projects.

Data collection is done automatically by using our bug history data collector tool. The data integration and validation is done by using Engineering Knowledge Base (EKB). By using OSMF to collect, integrate and validate the data, we have a clean data to be analyzed using ProM. The OSMF can make analysis on the integrated data easier than analysis on single separated data.

6.2 Evaluation of Engineering Process Measurement Results

The evaluation of actual engineering process model is done by checking its conformance to the designed process model. The process model that is generated based on Fedora bug history data is shown in Figure 4(A). The designed process model (Bugzilla Life Cycle) shown in Figure 4(B).

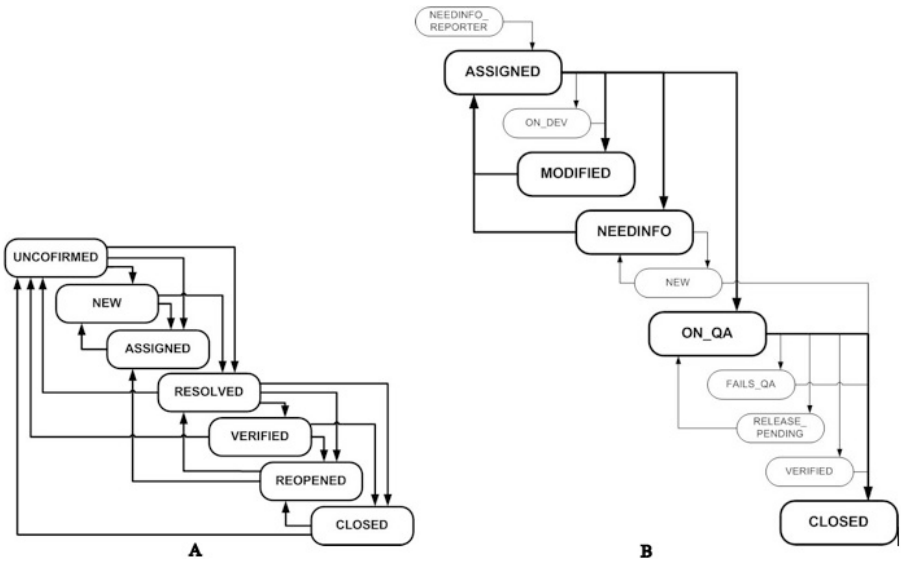

Fig. 3. Process Model of Bug Status. (A) Bugzilla LC. (B) Fedora 14

From the process model generation on RHEL and Fedora, we answer the two defined research hypotheses as follows:

RHEL and Fedora developers are following the naming and the ordering of the bug states from Bugzilla life cycle. As shown in table 1, we can see the differences of number of states used in the designed process model (Bugzilla life cycle) and states used in the generated process models from RHEL and Fedora. Therefore $\{\exists\psi \in (\text{RHEL,Fedora}) \mid P(\psi) \neq P(\phi)\}$ thus we can reject our null hypothesis H01.

An interpretation of these results can be the fact that the number of bug states available and used in RHEL and Fedora are different, meaning that both OSS projects are using different development strategy.

OSS projects developers are using all bug states for each bug history in the same number of frequency. As shown in table 2, each bug state is used in different frequency by the developers. Some bug states are used more often than the others. Therefore $\{\forall si,si+1 \mid N(si) \neq N(si+1)\}$ thus we can reject our null hypothesis H02.

An interpretation of these results can be the fact that the typically OSS projects do not follow a strict waterfall-like software engineering process, but rather a sometimes mixed dynamic software engineering process.

The proposed approach can be generalized to test other hypothesis and work on other kind of OSS project data, with some adaptations depend on the type of bug reporting tools used.

7 Summary and Future Work

Measurements of OSS processes are needed as an initial way to improve the quality of OSS processes and products. OSS project managers need to collect, integrate and analyze heterogeneous data originating from different tools used in the OSS project, such as source code management, developer's mailing list, and bug reporting tools to observe the processes and determine the status of OSS projects.

In this paper, we have explained the contribution of a measurement framework in improving the process quality in OSS projects. We used bug history data from Red Hat Enterprise Linux (RHEL) and Fedora projects as a use case for our measurement framework application and use the Heuristics Mining algorithm of the Process Mining tool ProM. The analysis results on conformance checking of process models from RHEL and Fedora bug history data can be used to improve the process quality.

Future Work. Future work will include the risk analysis on handling the bugs in the OSS project development in order to improve the quality of OSS products. Data sources that currently limited to bug report will be expanded to include other sources.

Acknowledgments. This work has been supported by the Christian Doppler Forschungsgesellschaft, the BMWFJ, Austria and the Ministry of Education, Republic of Indonesia.

References

1. Biffl, S., Sunindyo, W.D., Moser, T.: Semantic Integration of Heterogeneous Data Sources for Monitoring Frequent-Release Software Projects. In: International Conference on Complex, Intelligent and Software Intensive Systems, pp. 360–367. IEEE Computer Society (2010)
2. Crowston, K., Annabi, H., Howison, J.: Defining Open Source Software Project Success. In: 24th International Conference on Information Systems (2003)
3. Crowston, K., Li, Q., Wei, K., Eseryel, U.Y., Howison, J.: Selforganization of teams for free/libre open source software development. Inf. Softw. Technol. 49, 564–575 (2007)
4. Deshpande, A., Riehle, D.: The Total Growth of Open Source. Open Source Development, Communities and Quality, 197–209 (2008)
5. DiBona, C., Ockman, S., Stone, M.: Open Sources: Voices from the Open Source Revolution. O'Reilly Associates, Inc. (1999)
6. Kohlbacher, M.: The Effects of Process Orientation on Customer Satisfaction, Product Quality and Time-Based Performance. In: 29th International Conference of the Strategic Management Society (2009)
7. Mendling, J.: Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness. Springer, Heidelberg (2008)
8. Mockus, A., Fielding, R.T., Herbsleb, J.: A case study of open source software development: the Apache server. In: 22nd International Conference on Software Engineering. ACM, Limerick (2000)
9. Raymond, E.S.: The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary. O'Reilly & Associates, Inc. (2001)
10. Reijers, H., van Wijk, S., Mutschler, B., Leurs, M.: BPM in Practice: Who Is Doing What? In: Hull, R., Mendling, J., Tai, S. (eds.) BPM 2010. LNCS, vol. 6336, pp. 45–60. Springer, Heidelberg (2010)
11. Reijers, H.A.: Design and Control of Workflow Processes: Business Process Management for the Service Industry. Springer, Heidelberg (2003)
12. Rubin, V., Günther, C., van der Aalst, W., Kindler, E., van Dongen, B., Schäfer, W.: Process Mining Framework for Software Processes. In: Wang, Q., Pfahl, D., Raffo, D.M. (eds.) ICSP 2007. LNCS, vol. 4470, pp. 169–181. Springer, Heidelberg (2007)
13. Sharma, S., Sugumaran, V., Rajagopalan, B.: A framework for creating hybrid-open source software communities. Information Systems Journal 12, 7–25 (2002)
14. Trung, D.-T., Bieman, J.M.: Open source software development: a case study of FreeBSD. In: 10th International Symposium on Software Metrics, pp. 96–105 (2004)
15. van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M.: Business Process Management: A Survey. In: van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M. (eds.) BPM 2003. LNCS, vol. 2678, pp. 1–12. Springer, Heidelberg (2003)
16. vom Brocke, J.H., Rosemann, M.: Handbook on Business Process Management: Strategic Alignment, Governance, People and Culture. Springer, Berlin (2010)
17. Zur Muehlen, M.: Workflow-Based Process Controlling. Logos (2004)