Resolution Trend of Just-in-Time Requirements in Open Source Software Development

Tanmay Bhowmik* and Sandeep Reddivari†
* Dept. of Math., Computer Science and Information Systems, Northwest Missouri State University, Maryville, MO, USA
† School of Computing, University of North Florida, Jacksonville, FL, USA
bhowmik@nwmissouri.edu, sandeep.reddivari@unf.edu

***Abstract*—Research in "just-in-time" requirements engineering has recently emerged. Some research has explored the nature of just-in-time requirements analysis in open source software (OSS) systems. Whereas, others have focused on techniques, such as traceability-enabled refactoring and horizontal traceability, in order to help manage just-in-time requirements. Little is known, however, about the resolution trends of just-in-time requirements in OSS development. In this position paper, we analyze the resolution time of the requirements of Firefox and Mylyn, and identify interesting patterns throughout their development history. Our analysis instigates five intriguing questions regarding the characteristics of just-in-time requirements engineering for OSS systems, and opens further research avenues in this area.**

***Index Terms*—Just-in-time RE; open source RE, stakeholders' socio-technical interaction; social band of human action**

I. Introduction

Requirements engineering (RE) is a set of activities concerned with identifying and communicating the purpose of a software system, and the contexts in which it will be used [1]. Traditionally, RE has been considered as a centralized, collocated, and phase-specific process associated with individual projects or project components. Much of the traditional RE has focused on models and techniques in order to aid identification and documentation of stakeholders and their needs in a form that can be analyzed, communicated, agreed upon, and eventually realized and validated. However, in the agile and open source software (OSS) development environments, RE activities need to focus on generation, negotiation, adaptation, realization, and maintenance of requirements in an iterative, and dynamic software-intensive ecosystem [2]. RE activities in the OSS development paradigm are therefore no longer part of a centralized process specific to a particular phase of software development. Rather, these activities are performed during the overall software development process in a more ad-hoc and as needed manner.

Recent research has highlighted the ad-hoc and dynamic nature of RE activities in the OSS development paradigm [2], [3], [4], [5], [6], [7], [8]. In order to reflect the as needed nature of the requirements in OSS development, Ernst and Murphy [4] introduced the notion of "just-in-time" requirements indicating that the identification and realization of these requirements are rather tightly coupled. Just-in-time requirements are often captured as lightweight informalisms [3], such as user stories, and the developers constantly and iteratively elaborate and clarify the requirements during realization. Some research has explored the nature of just-in-time requirements analysis in OSS systems [3], [4], whereas others have focused on techniques, such as traceability-enabled refactoring [6] and horizontal traceability [6], in order to help manage just-in-time requirements. Little is known, however, about the resolution trends of just-in-time requirements that help addressing exact user-needs in a timely manner.

In this position paper, we report on an exploratory study [9] that investigates the just-in-time requirements of Firefox [1] and Mylyn [10], two large scale OSS systems successful in their application domains. In particular, we analyze the resolution time of the requirements of our subject systems starting from their inception, and identify interesting patterns throughout their development history. Our analysis instigates intriguing questions regarding the characteristics of just-in-time RE for OSS systems and opens further research avenues in this area. The rest of the paper is organized as follows. Section II covers background information. Section III details the study setup. Section IV presents data analysis and discussion, followed by Section V giving a summary of some related work. Finally, Section VI concludes the paper.

II. Background

The concept of "just-in-time" as a production strategy [11] can be traced back to the 1950s [11].[2] In order to meeting customer demand at the right time and in the exact amount, Toyota and other Japanese firms introduced just-in-time production strategy that followed three principles of economic growth: build only what is needed, eliminate anything which does not add value, and stop if something goes wrong [11].

In the domain of software development, Scacchi [3] pioneered the research in understanding the OSS requirements. Alspaugh and Scacchi [8] studied the RE activities of OSS systems and emphasized that many successful OSS projects do not follow the classical, one-time RE process. Scacchi [3] identified a set of twenty-odd different types of what he called 'software informalisms' in use across a wide variety of OSS projects. In a further study, Scacchi et al. [12] examined OSS systems from five different application domains and found that very often a new OSS requirement is a feature informally captured through a story telling or a user experience at the initial stage.

[1] http://www.mozilla.org/en-US/firefox/new/
[2] Call for paper, Just In Time RE Workshop, Ottawa, Canada, 2015

TABLE I
DATA COLLECTION OF SUBJECT SYSTEMS

System	Application domain	Analyzed history	# of requirements studied	# of code files	Written in
Firefox	Web browser	2003–2011	1,985	1,968 (C/C++)	C/C++, JavaScript
Mylyn	Eclipse plug-in	2005–2012	451	2,321	Java

The study presented in this position paper is built upon this pioneering work. In particular, we investigate the OSS requirements recorded in issue tracking systems from a resolution perspective, and draw discussion points in terms of research questions regarding just-in-time RE in OSS development.

III. STUDY SETUP

In our study, we analyze the successfully implemented requirements of two OSS systems: Firefox and Mylyn. We select these projects as the subject systems for our exploratory study [9] due to a number of reasons. First, they are large OSS systems and were previously studied in software engineering research [4], [10], [13]. Second, they are from different application domains, they are successful in their own domains, and therefore they can be considered representatives of their own application domains. Third, the relevant data required to conduct this study is freely available online in the Bugzilla issue tracking system. This enables other researchers to replicate our study. Next is a brief description of our chosen systems.

Firefox is a successful open source project and a dominating web browser since its first release in 2004. From November 2004 to June 2011, Mozilla released Firefox stable versions 1.0 through 5.0, and after that made some rapid releases.[3] We collect data about the requirements from the beginning of Firefox history until the release of version 5.0.

Mylyn is an Eclipse plug-in that monitors programmer activity in the Eclipse IDE [10]. It was first started as a part of a Ph.D. thesis supervised by Gail Murphy at the Software Practices Lab at UBC.[4] We consider the requirements of Mylyn from its starting in 2005 till February 2012.

For every successfully implemented requirement (i.e., closed requirement), we collect information as follows: the issue ID, the issue owner (i.e., the stakeholder to whom the issue is assigned), reporting time stamp, closing time stamp, and the number of comments and artifacts posted. All the information is directly available on the issue page. We run a web scraping tool written in Java to automatically collect the required information. Table I presents an overview of the collected data that we analyze for the two systems. Note that the number indicating the size of Firefox does not include JavaScript files. In what follows, we detail the analysis of the collected data and further discuss our observation.

IV. DATA ANALYSIS AND DISCUSSION

In this position paper, we hinge our discussion about the trends in just-in-time requirements resolution in OSS software development environment. To that end, we analyze the resolution time of the requirements of Firefox and Mylyn over their history and present further discussion based on our observation.

A. Data Analysis

For both Firefox and Mylyn, we calculate the resolution time for each successfully implemented requirement falling within our analysis history (see Table I). In calculating the resolution time, we take the difference between the issue closing time stamp and issue reporting time stamp, and measure the difference in hours. Note that in Bugzilla, every time stamp consists of a date and time in hours and minutes. Next, we order the requirements for each subject system in a chronological manner by sorting them based on their reporting time stamps.

Figure 1 and Figure 2 present the resolution time of the chronologically ordered requirements for Firefox and Mylyn respectively. The *x*-axis represents the unique issue ID designated by Bugzilla, whereas *y*-axis shows the resolution time in hours. Note that *x*-axis also indicates the time span of the analyzed history, i.e., 2003 to 2011 for Firefox and 2005 to 2012 for Mylyn, in an indirect manner. The specific points of major releases along the time span are indicated by arrows. The main objective of the *x*-axis, however, is to accommodate all the closed requirements. Thus, the space in the figures between two consecutive releases may not represent the time difference in a consistent scale. Rather it shows all the successfully completed requirements proposed in the corresponding time window.

B. Discussion

In Figure 1, we observe a very interesting and apparently consistent pattern in the resolution time of Firefox requirements. We notice that, in every release cycle, the majority of the requirements proposed earlier have relatively longer resolution time. However, the resolution time keeps decreasing as we move towards the release, and it is considerably low for the requirements proposed within a time period of 2 to 3 months preceding a release (cf. the encircled areas in Figure 1). The resolution time for Mylyn requirements, however, tells us a different story than that of Firefox. In Figure 2, we notice that Mylyn requirements are a combination of relatively short and long resolution times distributed over the development history. In other words, we do not observe any apparent pattern in the chronologically ordered requirements resolution time.

At this point, it is worth mentioning that Firefox and Mylyn vary in their governance structure [14] and task triage process [15]. Firefox is a *foundation* project and follows a *volunteer-based* triage process. Mylyn, though started in a *monarchy* way as part of Mik Kersten's Ph.D. thesis, gradually

[3]http://www.mozilla.org/en-US/firefox/releases/
[4]http://www.eclipse.org/mylyn/about/

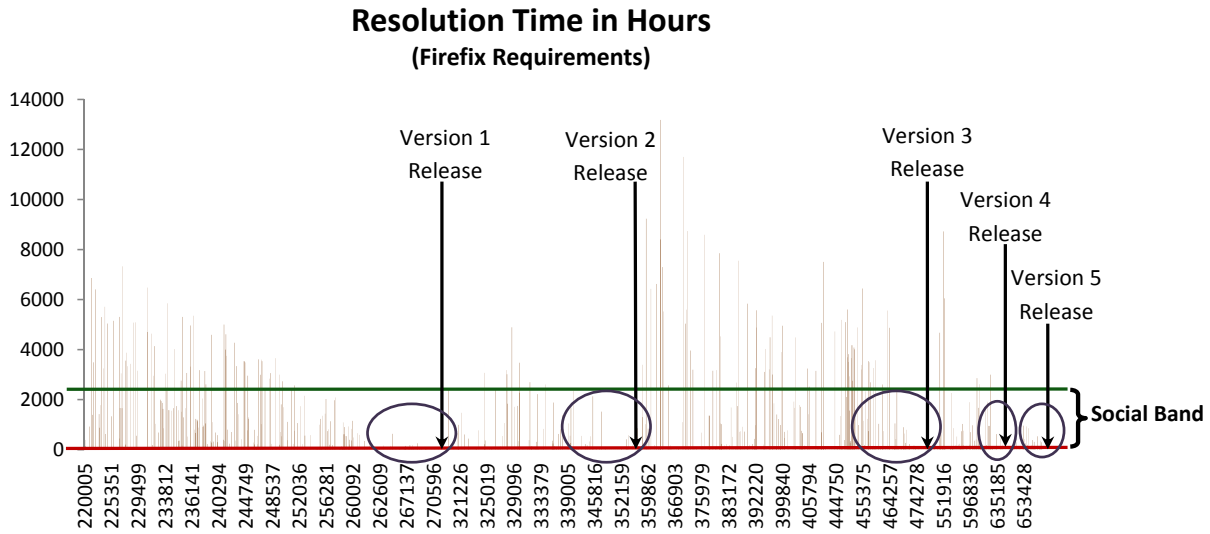

Fig. 1. Resolution time for Firefox requirements between 2003 and 2011.

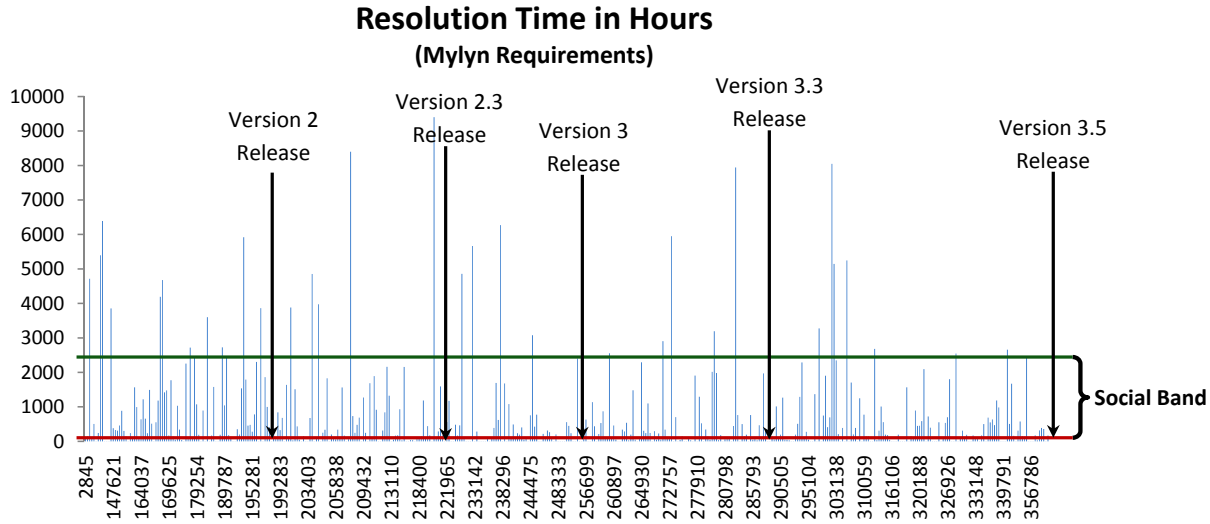

Fig. 2. Resolution time for Mylyn requirements between 2005 and 2012.

evolved into a *community* centering around the open source implementation of the task-focused interface. For Mylyn, determining the relevance and priority of each submitted issue is *developer-based*. Considering the variety in our subject systems along these different dimensions, the observations made so far inspire us to formulate the following questions regarding just-in-time RE.

- *Q1*: Is just-in-time requirements resolution in a traditionally volunteer-based OSS project highly release driven?
- *Q2*: In a traditionally volunteer-based OSS development environment, does just-in-time RE manifest itself more strongly when a release is approaching?
- *Q3*: Does the just-in-time requirements management of an OSS project that evolves around a central figure (e.g., Mik Kersten in Mylyn) follow a different approach than a traditionally volunteer-based OSS project?

In case of both Firefox and Mylyn, we notice another interesting phenomenon from the perspective of the time scale of human action [16]. The theories of cognition suggest that different human actions occur at different time bands. Depending on the time required to solve a problem by humans, Newell classified human actions into six different time bands: biological, cognitive, rational, social, historical, and evolutionary, where biological takes a few milliseconds and evolutionary taking millions of years [16]. For both the subject systems, we notice that many requirements fall within the social band of human action considering the resolution time (ranging from about 1.2 days or 28.8 hours to about 3.5 months or 2,520 hours [16]). In fact, about 90% of Mylyn Requirements fall into this category. In case of Firefox, almost all the requirements proposed within a 2 to 3 months time

period before a release (cf. the encircled areas in Figure 1) fall into the social band.

In a recently published study, we have demonstrated that higher socio-technical interaction [17] among stakeholders leads to improved developer productivity [18]. A further analysis on the collected data suggests that the average number of socio-technical interaction among stakeholders (in terms of posting comments and artifacts over the issue tracking systems [19]) for the requirements in the social band (19.5 and 13.2 for Firefox and Mylyn respectively) is higher than that of the overall average (18 for Firefox and 11 for Mylyn). Based on these observations, we formulate further questions regarding just-in-time RE.

- *Q4*: Does just-in-time RE predominantly fall within the social band of human action?
- *Q5*: Does just-in-time RE potentially lead to higher developer productivity?

Before we wrap up our discussion, an important aspect worth mentioning is that OSS development is largely a volunteer activity. In this software development paradigm, the stakeholders are motivated by their passion for sharing technological knowledge with others. Therefore, some theories from sociology and anthropology might be relevant to provide a strong foundation for investigating our research questions.

V. Related Work

In their seminal work, Ernst and Murphy [4] advanced our understanding about OSS RE by studying the requirements management of three successful OSS projects. Ernst and Murphy coined the notion of "just-in-time" requirements in OSS development indicating the fact that OSS requirements are captured in a rather informal manner and further elaborated during realization [4]. Another important finding from Ernst and Murphy's study is the dominant use of issue tracking systems like Bugzilla and Jira in managing OSS requirements [4].

In order to explore the possibility of a better tool support, Niu et al. [6] studied the requirements of both open source and proprietary software systems and introduced traceability-enabled refactoring for managing just-in-time requirements. Heck and Zaidman [7] studied open source feature requests in Subversion[5], and suggested a horizontal traceability technique using a Vector Space Model (VSM) in order to help realizing just-in-time requirements.

As research on just-in-time requirements has recently emerged [4], [6], [7], our knowledge on just-in-time RE for OSS systems is yet limited. In order to further expand the literature, in this position paper, we have studied the OSS requirements recorded in issue tracking systems from a resolution perspective, and drawn interesting discussion points regarding just-in-time RE in the OSS development paradigm.

VI. Limitation and Conclusion

In this position paper, we have analyzed the resolution time of the closed requirements for two large scale OSS systems and formulated five questions for further investigation. As the questions are derived based on our analysis on just two subject systems, further analysis involving more projects might be required to better identify more specific research questions.

Nevertheless, the questions presented in this paper instigate critical thinking, thereby serving as starting points for detailed research. Such research will help us gain valuable knowledge on just-in-time RE for OSS systems. In addition, it will provide important insights on the agile development paradigm for commercial projects.

[5]https://subversion.apache.org/

References

[1] B. Nuseibeh and S. Easterbrook, "Requirements engineering: a roadmap," in *Proceedings of the Conference on The Future of Software Engineering*, ser. ICSE '00, 2000, pp. 35–46.

[2] M. Jarke, P. Loucopoulos, K. Lyytinen, J. Mylopoulos, and W. Robinson, "The brave new world of design requirements," *Information Systems*, vol. 36, no. 7, pp. 992–1008, 2011.

[3] W. Scacchi, "Understanding the requirements for developing open source software systems," *IEE Software*, vol. 149, no. 1, pp. 24–39, 2002.

[4] N. A. Ernst and G. Murphy, "Case studies in just-in-time requirements analysis," in *Proceedings of the International Workshop on Empirical Requirements Engineering at RE*, 2012, pp. 25–32.

[5] M. Poppendieck and T. Poppendieck, *Lean software development: an agile toolkit*. Addison-Wesley Professional, 2003.

[6] N. Niu, T. Bhowmik, H. Liu, and Z. Niu, "Traceability-enabled refactoring for managing just-in-time requirements," in *Requirements Engineering Conference (RE), 2014 IEEE 22nd International*, 2014, pp. 133–142.

[7] P. Heck and A. Zaidman, "Horizontal traceability for just-in-time requirements: the case for open source feature requests," *Journal of Software: Evolution and Process*, vol. 26, no. 12, pp. 1280–1296, 2014.

[8] T. A. Alspaugh and W. Scacchi, "Ongoing software development without classical requirements," in *Proceedings of the International Conference on Requirements Engineering (RE)*, 2013, pp. 165–174.

[9] R. K. Yin, *Case study research: Design and methods*. SAGE Publications, 2008, vol. 5.

[10] M. Kersten and G. Murphy, "Mylar: A degree-of-interest model for ides," in *Proceedings of the International Conference on Aspect-Oriented Software Development (AOSD)*, 2005, pp. 159–168.

[11] T. Ōno, *Toyota production system: beyond large-scale production*. Productivity press, 1988.

[12] W. Scacchi, C. Jensen, J. Noll, and M. Elliott, "Multi-modal modeling of open source software requirements processes," in *Proceedings of the International Conference on Open Source Software*, 2005, pp. 1–8.

[13] S. Zaman, B. Adams, and A. E. Hassan, "Security versus performance bugs: A case study on firefox," in *Proceedings of the Working Conference on Mining Software Repositories (MSR)*, 2011, pp. 93–102.

[14] C. Bird, D. Pattison, R. D'Souza, V. Filkov, and P. Devanbu, "Latent social structure in open source projects," in *Proceedings of the ACM SIGSOFT International Symposium on Foundations of Software Engineering (SIGSOFT/FSE)*, 2008, pp. 24–35.

[15] J. Anvik and G. C. Murphy, "Reducing the effort of bug report triage: recommenders for development-oriented decisions," *ACM Transactions on Software Engineering and Methodology*, vol. 20, no. 3, Article No. 10, August 2011.

[16] A. Newell, *Unified Theories of Cognition*. Massachusetts: Harvard University Press, 1990.

[17] A. Meneely, B. Smith, and L. Williams, "iTrust electronic health care system: a case study," in *Software and Systems Traceability*, J. Cleland-Huang, O. Gotel, and A. Zisman, Eds. Springer, 2012.

[18] T. Bhowmik, N. Niu, W. Wang, J.-R. Cheng, L. Li, and X. Cao, "Optimal group size for software change tasks: A social information foraging perspective," *Cybernetics, IEEE Transactions on*, vol. PP, no. 99, pp. 1–12, 2015.

[19] T. Wolf, A. Schroter, D. Damian, and T. Nguyen, "Predicting build failures using social network analysis on developer communication," in *Proceedings of the International Conference on Software Engineering (ICSE)*, 2009, pp. 1–11.