The impact of formal QA practices on FLOSS communities – The case of Mozilla

Adina Barham
Hitotsubashi University, Graduate School of Social Sciences, 2-1 Naka,
Kunitachi, Tokyo, 186-8601, Japan
adina.barham@yahoo.com

Abstract. The number of FLOSS projects that include a QA step in the development model is increasing which suggests that a new layer may be emerging in the classic "onion model". This change might affect the information flow within projects and implicitly their sustainability. Communities, the essential resource of FLOSS projects, have been extensively studied but questions concerning QA remain. This paper takes a step towards answering such questions by analyzing QA mailing lists and issue tracker data for the Mozilla group of projects. Because the Bugzilla data set contains over half a million bugs, data processing and analysis is a considerable challenge for this research. The provisional conclusions are that QA activity may not be increasing steadily over time but is dependent on other factors and that the QA team and other groups of contributors form a highly connected network that doesn't contain isolates.

Keywords: quality assurance, test, Mozilla, social network analysis, information flow

1 Introduction

In recent years open source software has become a viable choice for a wider range of users, overcoming its initial status as a tool used only by experts and hackers. This phenomenon has led to higher expectations from end-users which translates into a greater need for responsible management, productivity over time, ease of maintenance, availability of support, increased quality and other features that now drive the success of FLOSS projects. This paper investigates whether and to what extent this change is affecting the way FLOSS communities develop software.

It is no longer a surprise when an open source project's community decides to adopt methodologies and policies that point more towards a hybrid development model than towards the bazaar model. This hybrid model combines development methodologies from traditional FLOSS development such as heavy community involvement, with those from proprietary development such as a QA phase comprising a series of elaborate steps taken to ensure a certain quality standard. Even

though QA practices are becoming more and more present in FLOSS projects, their success or failure depends greatly on actual community development [7], in other words on the project members. Furthermore, characteristics of the community such as its size [15] are important factors influencing the quality of a software product. We therefore need an up-to-date understanding of communities' structures and dynamics.

2 Background and Motivation

Open source software development has evolved substantially to keep up with the standards imposed by the continuously growing user base and the needs of the market. This implies refining the development process and pushing it towards a more sustainable model. But what does sustainability actually mean in this context? The Brundtland Commission's report defines sustainable development as development that "meets the needs of the present without compromising the ability of future generations to meet their own needs" [21]. In the context of open source software, this includes raising the quality standards of products by implementing more complex processes and rigorous methodologies. For example, it is safe to assume that as a project matures so does the testing process around it, which is a truism for both open source and proprietary software [11].

The importance of quality in open source is recognized as an important issue that needs to be further studied. This trend is illustrated by current research in the academic world [1-5], [13], [20] as well as research programs funded by various organizations and governments such as the Qualipso project [16] or Qualoss [10].

Another important trend in current research consists of analyzing the community that drives FLOSS using social network analysis. Although these studies focus on various aspects of the FLOSS community such as structure and dynamics [12], [14], communication patterns between core and periphery [17],[18] or migration within the hierarchy of FLOSS projects [9], none have sought to link QA with the rest of the community. This paper starts to fill that gap.

By analyzing the QA teams within one of the most famous FLOSS organizations (Mozilla) we can take a first step towards clarifying the position of QA within the open source community and further develop these findings into QA guidelines that can be applied to other FLOSS projects. Due to the particularities of each project there will not be a single recipe for success, but a study of this kind should provide important insights.

3 Research Questions

Q1: How does a QA contributor fit into the Mozilla community? Although recent research has defined more than three layers in the onion model [6, 9] it is generally accepted that a project's community can be split into: active users, co-developers and core developers. This research aims to investigate the extent to which QA is a step on the road from end-user to developer, or whether it has become established as a separate category of contributor.

Q2: What are the characteristics of QA activities within Mozilla? Members of the periphery also perform some QA tasks such as posting bugs on the issue tracker. It

has been noted that for the case of Firefox the percentage is 20 to 25% [18] and it would be interesting to compare and see the percentage of periphery involvement for other Mozilla products. Another aspect that should be investigated is how participants' activities evolve over time considering that QA tasks can vary based on technical difficulty. For example users may provide automated test tools, which might suggest that QA may be divided into two subgroups based on activity type.
Q3: What are the characteristics of communication patterns between QA members as well as with other project participants? The goal of analyzing the characteristics of communication patterns between QA members is to find the central figures within the community and observe their evolution over time as social networks have a continuously changing structure [8]. As previous research has shown, information access by community members correlates with productivity [19], and for this reason, interaction of QA with other layers of the Mozilla organization should not be ignored.

4 Data and Research Method

Mozilla has a QA phase in its development in the sense that community members form a layer that is responsible for the QA process and it is easily identifiable [22] (meaning that information associated with the QA team such as web pages, wikis, mailing lists, forums and so on can be easily found). For conducting this study, QA mailing lists and the issue tracker were analyzed using quantitative techniques and social network analysis (SNA).

Mozilla QA has two dedicated mailing lists, Mozilla.dev-quality and Mozmill developer, which is addressed to more technically aware users. A total of 3689 messages were exchanged (February 2006 – July 2011) between 327 distinct authors. More specifically 2535 e-mails were exchanged by 293 authors on Mozilla.dev-quality and 1155 e-mails were exchanged by 61 authors on Mozmill developer. As expected, the traffic and number of users is higher on the Mozilla.dev-quality mailing list due to the fact that it is less technical.

The issue tracker (Bugzilla) data set covers all Mozilla products since 1998 containing 687,221 bugs with 5,834,507 associated comments which brings up processing challenges due to its size. Bug ids range from 0 to 724,339 making a total of 724,339 where collected bugs represent 94.87% of the id range. The remaining 5.13% were not collected because they were not publicly available or due to bad html that could not be parsed.

Approximately 4400 distinct project members were identified as assigned to fix bugs. Without getting the data associated with code commits it is not safe to assume that these members were also the members that posted the bug fix, but it is safe to assume that they are code commiters. These users are also active when it comes to posting bug comments as well as sending e-mails on the QA mailing lists. After cross-referencing members active on the mailing lists and code commiters, 883 bugs were found most of which belonging to Firefox.

An interesting detail that can be noticed after analyzing the data in Table 1 is that most activity levels show a steady increase, which may indicate a growth in the community as well as an improvement in the information flow between layers of the community. This improvement is also suggested by the fact that members active on the mailing lists have bugs assigned to them.

Table 1. Activity levels on a yearly basis

	2006	2007	2008	2009	2010	2011
Comments	328846	335323	467087	528199	658030	703857
Bugs	42015	41995	56785	60880	78089	78896
E-mails	343	361	556	1307	739	384
Dev bugs	119571	123234	174742	177776	227123	226555
Dev Comments	258458	271679	375729	449539	541707	561853
Dev e-mails	196	286	343	953	500	264

If we consider that 11 e-mails (average number of e-mails sent) is the lower limit for highly active users then Pareto's principle is somewhat applicable in the sense that only 16.8% of the users send more than 11 e-mails and 17.69% of users receive more than 11 replies. Following the same principle, only 4.39% of users show a higher than average activity posting more than 39 comments and 9.25% more than 6 bugs. From all the e-mails exchanged, 152 (4.12%) were sent by authors that had sent only one e-mail throughout the period taken into consideration for this research. On the other hand, 135466 bugs (19.70%) were posted by members that had posted only one bug throughout the period taken into consideration. Firefox was the Mozilla product with most of these "hit and run" bugs.

In this phase of the research, due to the fact that data collection and cleaning took longer than anticipated, social network analysis techniques could not be applied to the whole data set. Instead interaction was analyzed between active members on the mailing list (more than 10 e-mails sent – 55 users) and 10 members fairly active on the issue tracker. The resulting network does not depict relations between all QA members and its role is only to offer a sample of the interaction patterns within the community. After eliminating loops (replies to themselves) this sub-network had a number of 1433 participants with 2593 connections; 933 of these connections were formed by more than one interaction. The average degree is 3.16, which means that the average number of connections a member has is approximately 3.

5 Conclusions

Q1: How does a QA contributor fit into the Mozilla community? Considering the fact that the Mozilla QA team has dedicated communication channels, one can draw the conclusion that it represents a separate layer in the community model. Although, at this point of the research a clear definition of the tasks performed by QA members has not been made, evidence such as the existence of a QA mailing list oriented to more technically aware users might suggest that there is more than one type of QA task.

Q2: What are the characteristics of QA activities within Mozilla? As expected the activity of members of the community that "hit and run" (open one bug and never contribute again, send one e-mail and never contribute again) is higher on the Issue Tracker than on the QA mailing list. This may suggest that QA mailing list members have a more sustained activity in the Mozilla community. Another difference is that issue tracker activity has shown an increase over time while mailing list data showed

a peak level. This might suggest that mailing list activity may not be related to time progression but to other variables that need to be found. On the other hand, the increase in activity on the issue tracker points out the community has grown over the years.

Q3: What are the characteristics of communication patterns between QA members as well as with other project participants? Data used for the social network analysis section of this study was performed only on a sample due to time related issues and thus a general conclusion regarding communication patterns can't be drawn at this point. However, the sample shows no small groups of people working together but a team spanning both mailing lists and issue tracker. In addition, judging by the activity of QA members and code commiters on the issue tracker it is safe to say that interaction with other community members has been increasing. This suggests that it is unlikely that there will be participants that control the flow of information, or bridges between the QA team and other layers of the community.

6 Limitations and Further research

The purpose of this study is to create a precedent for further research in this direction in order to come up with general guidelines that can be applied on a wider scale. It is logical to conclude that by analyzing the structure and behavior of only Mozilla QA, one can't obtain a foolproof method to successfully implement QA practices due to the variety and uniqueness of every FLOSS project. In addition, community members might also use other communication channels that are not publicly available. This is one reason why findings should be confirmed with a qualitative follow-up. Another reason to go back to the community is to correlate data peaks and other anomalies with actual situations.

In the next phase, social network analysis will be applied to the whole data set using time frames and with consideration to time decay affecting connections between members of the community. Furthermore, in order to obtain an objective categorization of community members it is necessary to integrate previously acquired results with code comment data. It is essential to separate the QA members from developers and track their evolution within the community by monitoring their activity levels within different time frames and in different environments.

Whether the quality of Mozilla products have improved or not after the introduction of a formal QA step could represent a valuable assessment for other growing FLOSS communities. For this reason further phases should also include quality evaluation and measurement of Mozilla products as well as a classification and definition of QA procedures within Mozilla.

7 References

1. Halloran, T. J., & Scherlis, W. L.: High quality and open source software practices. In: 2nd Workshop on Open Source Software Engineering (2002)
2. Hedberg, H., Iivari, N., Rajanen, M., & Harjumaa, L.: Assuring Quality and Usability in Open Source Software Development. In: First International Workshop on Emerging Trends in FLOSS Research and Development, FLOSS'07., pp. 2-2 (2007)

3. Michlmayr, M., Hunt, F., & Probert, D.: Quality practices and problems in free software projects. In: Proceedings of the First International Conference on Open Source Systems, pp. 24–28, (2005)
4. Schmidt, D. C., & Porter, A.: Leveraging open-source communities to improve the quality & performance of open-source software. In: Proceedings of the 1st Workshop on Open Source Software Engineering (2001)
5. Chengalur-Smith I., Sidorova A., Daniel S.: Sustainability of Free/Libre Open Source Projects: A Longitudinal Study. In: JAIS 11 (2001)
6. Wiggins, A., Howison, J. & Crowston, K.: Social dynamics of FLOSS team communication across channels. In: Proceedings of the Fourth International Conference on Open Source Systems (2008)
7. Kilamo, T.: Essential Properties of Open Development Communities. In: Proceedings of the OSS 2011 Doctoral Consortium (2011).
8. Watts, D. J., A.: Twenty-first century science. In: Nature, Vol. 445, No. 7127, pp. 489-489 (2007)
9. Jensen, C., & Scachi, W.: Role Migration and Advancement Processes in OSSD Projects: A Comparative Case Study. In: Proceedings of the 29th international conference on Software Engineering , pp.364-374 (2007).
10. Quality in Open Source Software, http: //www.qualoss.org/
11. DiBona C., Cooper D., Cooper M.: Open Sources 2.0: The Continuing Evolution. O'Reilly, USA (2006)
12. Crowston K., Howison, J.: The social structure of Free and Open Source software. In: First Monday, Vol. 10, No. 2 (2004)
13. Spinellis D., Gousios G., Karakoidas V., Louridas P., Adams P. J, Samoladas I., Stamelos I.: Evaluating the Quality of Open Source Software. In: Electronic Notes in Theoretical Computer Science, Volume 233, Proceedings of the International Workshop on Software Quality and Maintainability (2009)
14. Mockus A., Fielding R. T, and Herbsleb J. D.: Two Case Studies Of Open Source Software Development: Apache And Mozilla. In: ACM Transactions on Software Engineering and Methodology, volume 11, number 3, pp. 309–346 (2002).
15. Sowe S., Ghosh R., & Haaland K.: A Multi-Repository Approach to Study the Topology of Open Source Bugs Communities: Implications for Software and Code Quality. In: 3rd IEEE International Conference on Information management and engineering, IEEE ICIME (2011).
16. Qualipso (Trust and Quality in Open Source Systems), http://www.qualipso.org/
17. Oezbek C., Prechelt, L. & Thiel F.: The Onion has Cancer: Some Social Network Analysis Visualizations of Open Source Project Communication. In: Psychology, Section 4, 4-9, (2010).
18. Masmoudi H., den Besten M. L., De Loupy C. and Dalle J. M.: 'Peeling the Onion': The Words and Actions that Distinguish Core from Periphery in Bug Reports and How Core and Periphery Interact Together. In: Fifth International Conference on Open Source Systems (2009)
19. Aral, S., Brynjolfsson, E. & Van Alstyne, M.: Productivity Effects of Information Diffusion in E-mail Networks. In: Proceedings of ICIS 2007 (2007)
20. Aberdour M.: Achieving Quality in Open Source Software. In: IEEE Software, pp. 58-64 (2007)
21. Burtland Comission: The Bruntland Report. United Nations, (1987)
22. QMO, https://quality.mozilla.org/