Channeling Firefox Developers: Mom and Dad Aren't Happy Yet*

Jean-Michel Dalle[1], Matthijs den Besten[2], and Héla Masmoudi[1]

[1] Université Pierre et Marie Curie, Paris, France;
jean-michel.dalle@upmc.fr,masmoudi_hela@yahoo.fr
[2] University of Oxford, Oxford, UK; matthijs.denbesten@oerc.ox.ac.uk

Summary. Firefox, a browser targeted at mainstream users, has been one of the big successes of open source development in recent years. That Firefox succeeded where earlier attempts failed is undoubtedly due to the particular choices that were made in the process of development. In this paper, we look at this process in more detail. Mining bug reports and feature requests related to Firefox in Mozilla's Bugzilla bug tracker system, we find that the attention developers devoted to reports and requests was influenced by several factors. Most importantly, other things being equal, reports and requests from outsiders increasingly tend to be ignored. While such behavior may have helped to shield Firefox from the "alpha-geek power user" in the early stages of development, it also makes it difficult for "mom and dad" to let their voice be heard even after they have adopted Firefox.

1 Introduction

In June 2006, Blake Ross, one of the initiators of Firefox, gave an interview to Olivia Ryan. At some point, the conversation turned to the issue of project management [5]:

> Olivia Ryan: And so was it difficult sometimes to strike that balance between working on an open project and trying to keep end-users in mind? Blake Ross: Yes. Yes. Everybody hated us for a long time.

Everybody may have hated Blake Ross and his companions, but they succeeded: In 2004 Firefox appeared and it quickly became, as an article in Wired put it, "the hot new browser rocking the software world." What we would like to know now is how they succeeded.

Understanding the processes that helped shape Firefox is important for a variety of people. First of all, as more and more users adopt Firefox as their default browser, they need to be reassured that Firefox remains a stable and secure alternative that is geared towards their needs. But also the stakeholders

* The work presented here has benefited from discussions with Paul A. David and others in the context of the OII/MTI project on Distributed Problem Solving Networks.

Please use the following format when citing this chapter:

Dalle, J.-M., den Besten, M. and Masmoudi, H., 2008, in IFIP International Federation for Information Processing, Volume 275; *Open Source Development, Communities and Quality*; Barbara Russo, Ernesto Damiani, Scott Hissam, Björn Lundell, Giancarlo Succi; (Boston: Springer), pp. 265–271.

of companies that have invested in Firefox, or that consider participation in similar projects, need to ascertain whether the Firefox approach to open source development works. Ultimately, even the developers themselves, who may not always be aware of the processes that channel their behavior, could benefit from our exercise in data mining.

Methodologically, we subscribe to the suggestions of Scacchi and Sack and others [7, 6] that the processes of open source software development can be discovered by studying the patterns of interactions between the developers and the resulting code. Our recurring obsession, in this study and in previous studies, is to identify what one could call *stigmergic* features in the interactions [1]. Like pheromone guiding ants from the nest to the food source, we consider that, lacking strict managerial control, there must be elements in the code and documentation that nudge developers to devote their attention in specific directions. Previous results of ours suggest that developers do indeed react to signals like the complexity of the code and address *complex* tasks in teams [3]. Further, we found that *contextual* elements, be it the level of detail in the specification of the task or linkage with related tasks, affect the speed with which tasks are dealt with and we also found that a lack of contextual elements may make it necessary to revisit the task several times [2]. In the study here, we focus at the attraction of ownership: Does it make a difference whether the task stems from the project *core* or from its *periphery*? In particular, does it make a difference for the resolution of a bug, whether the bug was reported by an insider or by an outsider? Short answer: yes. The long-term research agenda to which this study contributes is to derive an ontology of distributed process management from the structured, semi-structured, and unstructured data that document open source software development. Our hope for our ontology is that it could help assess the sustainability and evolution of projects like Firefox.

“We’re making a product for mom and dad.” Blake Ross said of Firefox [4]. That is probably not the case for our research. Nevertheless, in what follows, we will make a valiant attempt to share our preliminary findings. First, we describe in greater detail the bug-report data that we collected and how we connected them with information on bug-histories and code-patches. Next, we will present our preliminary results which suggest that the same process that helped make Firefox a product for mom and dad in the first place now threaten to lead to a neglect of mom and dad’s needs in the future.

2 Data

Sack and others note that developers coordinate their activity almost exclusively in three information spaces: “the implementation space, the documentation space, and the discussion space” [6]. Our collection covers each of these spaces, at least partially. It covers the implementation space in so far as we only consider bugs that are mentioned in commits to the official code base that were kept by the Mozilla concurrent versions system (CVS). For those bugs, in so far as they are associated with a file of which at least one revision is part of a Firefox release, we retrieve the bug-reports and the logs of changes to those bug reports kept by the bug tracker system Bugzilla. Thus, we are covering the documentation space as well. We also cover the discussion space since mailing-list style comments relating to the bugs are included in the reports. However, if Blake Ross is right in his assessment that Internet Relay Chat "was a big means of communication and especially with Firefox early on" [5], then we might have a problem, because, too our knowledge, archives of those chats have not been kept.

On the other hand, so far, we haven't done the text mining to which the chat archives and bug comments lend themselves. Instead, for now we have focused on a range of more readily quantifiable indicators. From the CVS, we retrieved for each bug the number of different authors referring to the bug in their commit-comments; the number of files touched by these commits; the number of distinct comments and the number of commit; and, finally, the number of lines of code added and deleted through the commits. From the bug-reports, we retrieved the number of persons copied in the bug resolution discussion; the number of other bugs the bug depends on or blocks; the numbers of attachments, patches and comments and the number of distinct contributors to the discussion; and, besides, the number of bugs that were identified as a duplicate of the bug reported. In addition, we retrieved the priority assigned and the severity estimated for each bug. With help of the change-log of the bug-reports, we also retrieved the time that passed between the opening of the report and the assignment of someone in charge of the resolution process; we retrieved the number of times the bug was (re-)assigned whether its priority was incremented or decremented or had been changed more than once; whether its severity increased or decreased or changed more than once — where the importance was judged to be in order of trivial, enhancement, minor, normal, major, critical, blocker. We determined whether the initial status of the bug report was New or Unconfirmed; whether the bug was reopened at least once; the number of report edits by the bug-reporter, and the number of report edits by the last person in charge of its resolution.

We drew two samples of bugs from the 40 000 or so that have resulted in a change to the Firefox code base. The first from the early stages of the project, when Firefox was still called Phoenix, in between release 0.1 and 0.5; the second from a later, more mature, phase between release 1.5 and release 2.0. For the moment, we ignore bugs where the severity was judged to be enhancement as we want to focus on bug reports rather than feature requests.

3 Results

Blake Ross described the Firefox development philosophy thus [4]:

> We're making a product for mom and dad. You still have a voice here, but some of the features that you think we should add may not be the

ones that they want tot use. So you have to take our word for it that, even though 500 of you want something right now, you may actually be in the minority of a much larger group that we're pursuing that's going to be silent during this phase of development.

Bug reports provide an invaluable window where we should be able to see this philosophy in action. It is there that we can witness the interaction between users and developers and it is also there that we can see how developers manage to balance the demands from the early adopters, the "alpha-geek power user" against those from those targeted, mom and dad. Take for instance bug 213186[2]. This bug report is a request to alter the text in the preference pane and in particular to replace the geek-humour of "Cookies are delicious delicacies" with a more appropriate explanation for concerned users without prior knowledge of the concept. Blake Ross himself was responsible for the "delicious delicacies" joke in the first place and his mom and dad might well have approved. Many other users however might be put off by the lack of information provided by people who seriously want to consider whether want to accept cookies or not. And the eventual resolution of bug 213186 shows that the concerns of these people were eventually recognized as "delicious delicacies" was replaced by "pieces of information stored by web pages on your computer[. . .]." Table 1 gives a few indicators for the extent and the duration of the bug-report. From these indicators, it is already clear that considerable effort was required to resolve this bug — effort that was mainly devoted to the construction of consensus between the geeks who like this humour and the advocates of mom and dad who did not.

Table 1. Characterization of bug 213186–*Please remove 'Cookies are delicious delicacies' from Options→Privacy→Cookies* and bug 171349–*Mozilla Firefox Icon is Windowing System's Standard Icon*.

Parameter Description	Bug 213186	Bug 171349
Reported	2003-07-19	2002-09-28
Fixed	2004-12-03	2005-05-30
Initial Status	Unconfirmed	New
Severity	trivial	normal
Time to First Assignment(*hrs*)	314	576
Number of Comments	55	206
Number of Authors of Comments	18	89
Number of Duplicates	1	31
Number of Patches	9	11

Bug 171349[3] is the other bug that is characterized in the table. The subject matter is less exciting — a lot of the attention for bug 213186 could be because

[2] `https://bugzilla.mozilla.org/show_bug.cgi?id=213186`

[3] `https://bugzilla.mozilla.org/show_bug.cgi?id=171349`

it is a bike shed like discussion in which it is easy to have an opinion[4] — but that does not make this bug less important *per se*. What is interesting about this bug is why it took so long to resolve. As one commenter remarks:

> > This bug was first discovered about 2 1/2 years ago and is still listed as "NEW"?
> Mozilla, like all the other big projects, has a LOT of bureuacracy [sic] going on. Ireported this so long ago I forgot all about it, and probably would've submitted it again if it weren't for the recent activity of my report being marked as a dupe... But even if it requires 5 people to check it and give it their OK,surely adding a .ico to the package can't be /that/ hard, can it? (Shish, 16/03/2005, comment #165)

As bug 171349 is the kind of bug that typically matters more to mom and dad than to a power user, the question is relevant whether the difficulty of the resolution of this bug was typical in Firefox development or the exception.

In order to get an impression of the general patterns of bug resolution in Firefox, we performed a survival analysis of a variety of samples of bug-reports related to Firefox. Table 2 presents the results. We carried out regressions on six samples: A sample of bugs related to Firefox in the early phases of its development; a sample of bugs related to Firefox in a later phase of its development; and for both of these samples a sub-sample of bugs started their life as New and another sub-sample for those bugs that started their life as Unconfirmed.

The difference between New and Unconfirmed is due to the fact that only a subset of the members of the Firefox community have the so-called CanConfirm privilege, according to which their bugs start as New and do not need confirmation — which is precisely needed otherwise to move from Unconfirmed to New. This privilege is granted by cooptation, based on the past track record of developers asking for it, and we therefore consider for the sake of the analysis here that it characterizes members of the core of the community, as compared to more peripheral members without this privilege.

What emerges from the table is that bug treatment in the early phases of Firefox was different from treatment at a later stage. More in particular, the difference between the treatment of bugs that stem from outsiders in contrast to the treatment of bugs in general seems more pronounced in the more recent samples. Besides, issues like the complexity of the problem and effort devoted to the contextualisation of the problem still play a role in determining the speed with which a bug is typically resolved.

4 Conclusion

Firefox is an open source project that so far has enjoyed a wide appeal among mainstream users as well as developers. In this paper, we argued that a study

[4] see `http://www.bikeshed.org`

Table 2. Significance and impact of variables controlling for bug fixing regimes (Survival analysis; Weibull fit).

Sample:	Phoenix (< 0.5)			Firefox (> 1.5)		
Parameter Description	All	New	Unconf'ed	All	New	Unconf.
# Bugs	7596	6200	1366	4721	3465	1256
Intercept	+***	+*	+**	+***	+*	+**
Number Of Different Committers	+***	+***	+***	+***	+***	
Number Of Files Touched in Codebase	+***	+***	+***	+***	+**	+•
Number Of Different Comments in Commits				+***	+***	
Number Of Commits	−*		−***	−***	−*	
Number Of Lines Of Code Added	−***	−***	+***			
Number Of Lines Of Code Deleted			+*			
Number Of Persons Copied	−**	−**			+*	−***
Number Of Attachments	+*		+*			
Number Of Patches				+•		
Depends On How Many Other Bugs	+**	+*	+*			+•
Blocks How Many Other Bugs	+*	+**				
Number Of Comments			•			+**
Number Of Authors Of Comments	+***	+***	+***	+***	+***	+***
Severity Trivial	+***	+***	+***	+***	+***	
Severity Minor	+***	+***	+***	+***	+***	
Severity Normal	+***	+***	+***	+***	+***	
Severity Major	+***	+***	+***		+*	
Severity Critical	+***	+***	+***		+**	−*
No Priority	−*	−*				
Priority 1	−**	−*	−*			
Priority 2	−*					+*
Priority 3	−*	−*				+•
Priority 4						
Log Time To First Assignment	+***	+***	+***	+***	+***	+***
Number Of Times Bug Was Assigned	+***	+***	+***	+***	+***	+***
Priority Was Increased	+*	•				
Priority Was Decreased	+**	+*	•	+**	+**	
Priority Was Changed More Than Once	•	•				+•
Severity Was Increased				+**	+***	
Severity Was Decreased				+*		+*
Severity Was Changed More Than Once						
Initial Status Was New				−**		
Bug Was Reopened At Least Once	•					−•
Number Of Edits By Bug Reporter			•	−**	−**	−**
Number Of Edits By Last Assignee		+***	•		+*	
Number Of Duplicates		−*	−*	−***	−***	

of the Firefox bug resolution processes is crucial to an understanding of the dynamics of its development. We described bug-reports and their association with code-commits. We described two bugs in detail and we discusses our preliminary efforts to detects patterns in the survival of bugs in general. Our findings suggest that Firefox developers tend to ignore the concerns that are voiced by outsiders, especially so in the later stages of its development. Does this mean that mom and dad, for whom Firefox was meant, are increasingly ignored as well — that remains to be seen. If this were true, however, it wouldn't bode well for Firefox' future.

The research described here constitutes just the first steps of a more ambitious research agenda in which the ultimate goal is to derive an ontology of distributed project management. What we have done here is to investigate the interaction between core and periphery in one project. For now, all we have is a suggestion that the current situation might not be optimal. But as our understanding grows and as our ontology expands, we could imagine a future in which the ontology itself would be used to optimize the development processes of open source software by accounting for the management of attention. And that could be a further step in making mom and dad happier.

References

1. J.-M. Dalle and P. David. Simulating code growth in libre (open-source) mode. In N. Curien and E. Brousseau, editors, *The Economics of the Internet.* Cambridge University Press, Cambridge, UK, 2007.
2. J.-M. Dalle and M. den Besten. Different bug fixing regimes? A preliminary case for superbugs. In *Proceedings of the Third International Conference on Open Source Systems*, Limerick, Ireland, June 2007. To appear.
3. M. den Besten, J.-M. Dalle, and F. Galia. Collaborative maintenance in large open-source projects. In E. Damiani, B. Fitzgerald, W. Scacchi, M. Scotto, and G. Succi, editors, *Open Source Systems*, volume 203 of *IFIP International Federation for Information Processing*, pages 233–244, Boston, 2006. Springer. Received best paper award.
4. J. Livingston. Blake Ross; creator, Firefox. In *Founders at Work: Stories of Startups' Early Days.* Apress, 2007.
5. O. Ryan. Interview with Blake Ross. Archived at http://mozillamemory.org, June 2006.
6. W. Sack, F. Détienne, N. Duchenaut, J.-M. Burkhardt, D. Mahedran, and F. Barcellini. A methodological framework for socio-cognitive analyses of collaborative design of open source software. *Computer Supported Cooperative Work*, 15(2-3):229–250, 2006.
7. W. Scacchi. Socio-technical interaction networks in free/open source software development processes. In S. T. Acuna and N. Juristo, editors, *Software Process Modeling*, pages 1–27. Springer, New York, NY, 2005.