Voting for bugs in Firefox: a voice for Mom and Dad?

Jean-Michel Dalle, Matthijs Den-Besten

Voting for bugs in Firefox: A voice for Mom and Dad?

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

[1] Université Pierre et Marie Curie, Paris jean-michel.dalle@upmc.fr
[2] Ecole Polytechnique, Paris matthijs.den-besten@polytechnique.edu

Abstract. In this paper, we present preliminary evidence suggesting that the voting mechanism implemented by the open-source Firefox community is a means to provide a supplementary voice to mainstream users. This evidence is drawn from a sample of bug-reports and from information on voters both found within the bug-tracking system (Bugzilla) for Firefox. Although voting is known to be a relatively common feature within the governance structure of many open-source communities, our paper suggests that it also plays a role as a bridge between the mainstream users in the periphery of the community and developers at the core: voters who do not participate in other activities within the community, the more peripheral, tend to vote for the more user-oriented Firefox module; moreover, bugs declared and first patched by members of the periphery and bug rather solved in "I" mode tend to receive more votes; meanwhile, more votes are associated with an increased involvement of core members of the community in the provision of patches, quite possibly as a consequence of the increased efforts and attention that the highly voted bugs attract from the core.

1 Introduction

Firefox is an open source project that explicitly tries to cater to the needs of a mainstream audience. Judging by its market share, as estimated for instance by the firm StatCounter (http://gs.statcounter.com), Firefox is succeeding. What might explain this success? To a large extent it is due to the leadership of people like Blake Ross, Dave Hyatt and Asa Dotzler and to their apparently correct judgement in deciding which features and bugs deserve the highest priority for development. In addition, however, we expect that the success is due to explicit mechanisms that have been put in place in order to make sure that the needs of mainstream users are assessed and addressed correctly. Voting for bugs in the Bugzilla bug tracking system is one such mechanism.

Voting for bugs is of course not unique to Firefox. It is a feature of the Bugzilla bug tracking system that has been activated by many of the projects who use it. Moreover, it is generally assumed that some sort of voting is a standard element of the open source software development model (see, e.g., [11]). Yet, apart from the governance of Apache [10] and Debian [12], there

are to our knowledge surprisingly few *explicit* analyses or even descriptions of voting procedures in free/libre open source software communities.

In her analysis of the emergence of voting procedures in Debian, O'Mahony focuses on the greater efficiency and transparency that is provided by voting compared to consensus based collective decision making usually dominant in small groups and presents the introduction of voting as a reaction to the increases in scale and scope of the Debian enterprise. Voting can however be useful even in small groups: see for example the seminal paper on collective problem identification and planning by Delbecq and Van De Ven [4], who propose a model of an effective group process in which the first phase of problem exploration is concluded with a vote. This, they note, serves to make the results of the exploration explicit and create pressure for change on the people who will be responsible for resolving the problem. Here voting is not just about representation, it is also about "getting heard".

In this context it might be useful to ponder the framework that Hirschman [5] developed on the means by which patrons can influence their organizations. Patrons have basically two options: either they leave the organization and buy or produce with a competitor ("exit"), or they express their concerns more explicitly through channels that might be provided by the organization ("voice"). In open source software development, the primary mechanism for "exit" is forking [9] and the primary mechanism for "voice" is to "show code" by proposing patches [1]. For mainstream users such as "mom and dad", however, neither forking nor patching is a realistic option, as they typically do not have the skills to do either. Since Firefox precisely wants to be a product for mom and dad [6], other options have to be found: "exit" can be implemented by switching to other browsers like Internet Explorer, Safari, Opera, and Chrome. For "voice", voting might be an appropriate answer.

In what follows, we present circumstantial evidence to support our conclusion that voting for bugs in Firefox is a means to provide a voice to mainstream users. This evidence is drawn from a sample of bug-reports maintained by Mozilla's Bugzilla and from explicit information on voters found at the same site. We present results regarding voters and bugs in sections 4 and 5, respectively. We present some more background information on our research in section 2 and describe our sampling strategy in section 3.

2 Background

This paper is number four in a succession of papers that we have presented at OSS conferences. In the first paper, presented in Limerick in 2007 [2], we analyzed bug reports from Mozilla and found that there are some bugs that take exceedingly long to resolve and that part of the reason for the existence of these "superbugs", as we named them, could be related to the insufficient provision of contextual elements such as screenshots in the bug threads. The second paper, presented in Milan in 2008 [3], analyzed bug reports that are associated with

the Firefox branch and exploited the existence of the so-called "CanConfirm" privilege — the privilege to declare bugs as "NEW" given that the initial status of bugs is "UNCONFIRMED" by default — to distinguish between core and peripheral participants within bug resolution processes and inquired whether various variables could influence the speed at which NEW and UNCONFIRMED bugs were patched. In the third and latest paper, presented in Skövde in 2009 [8], we used text mining techniques to arrive at a further characterization of participants within the core and the periphery of the Firefox community, stressing notably the fact that members of the core tend to use to pronoun "We" disproportionately while members of the periphery, conversely, seem to prefer to use "I".

By focusing on voters this paper tries to exploring an additional layer of the onion model of the Firefox community. To echo the language we used in our 2008 paper, while the likes of Blake Ross and Dave Hyatt clearly belong to the inner core, the periphery is probably mostly populated by "alpha-geek" users — that is, exactly by those people who Ross and Hyatt professed to ignore in Firefox development [6]. "Mom and dad" are not there. If at all, they are more likely to be among the *outer* periphery of people who are simply voting and contributing very little otherwise.

This line of inquiry obviously owes a lot to the pioneering work by Mockus et al. [10] and also to subsequent work by Ripoche [13], who established bug reports as an object of study. Together with others, e.g. [14], we continue to exploit this extremely rich source of information. Conceptually we have been inspired by the analysis of MacCormack at al. [7], who argue that it was necessary to make the existing code that was left by Netscape more modular before Mozilla could attract patches from the periphery. We wonder here whether votes could constitute another element which could help to explain how Firefox could turn the unwieldy open source project that Mozilla had become into the sleek browser for the mainstream market that we have now.

3 Sampling Method

For our analysis we constructed two types of data-sets. The first type relates to bugs, which may or may not have attracted votes, and concerns the history of the bug resolution process as it is recorded by Bugzilla as well information about the bugs that can be found in the logs of the CVS code repository. The second type concerns information that is stored in Bugzilla about the activities of the people who have voted for one or more bug. For each type we have obtained several sets and sub-sets of data based on three criteria: first of all, we are interested in bugs for which we can assume that they have had an impact on the Firefox code-base in the sense that they are mentioned in the commit-comments and the CVS repository; secondly, we are interested in bugs that have received votes; and finally, we are interested in bugs that Bugzilla associates with the Firefox project. Similarly, we are interested in people who voted for

bugs which eventually found their way into the CVS; we are also interested in the other bugs they voted for; and we are interested in the people who voted with them on Firefox bugs. Figure 1 is a Venn diagram which illustrates this admittedly somewhat complex constellation of sets. Our main interest is in the intersection of bugs from CVS that have been voted on and are also officially associated with Firefox as well as in the people who have voted for this particular set of bugs. Other than that we also have some interest in the other subsets formed by the intersection of the three criteria in order to be able to compare and contrast with what we find in our main set.

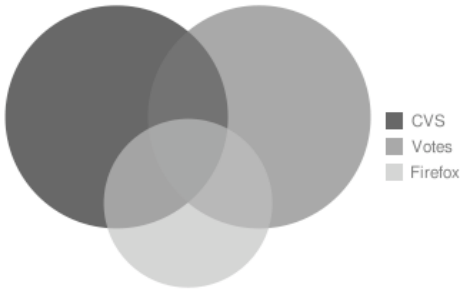

Fig. 1. Sampling of bugs: 37408 appear in the code base (CVS); among them 3787 have attracted at least one vote (Votes ∩ CVS) and 418 among these are associated with the Firefox project (Firefox ∩ Votes ∩ CVS).

In practice we followed a procedure similar to snowball sampling in order to obtain our sets and subsets of bugs and voters. We started with the bugs that we found in the logs of our copy of the Mozilla's CVS archive concerning the code for Firefox up until version 2.0. Only a small subset of these bugs, some 10%, has ever attracted a vote. Associated with those bugs is a list of all people who have cast a vote (people can also retract there vote later; in that case they fall through our net). Our next step was to retrieve all those lists and compile a list of CVS voters constituted by union of all sets of voters contained in these lists — we found 11826 of them. The list of voters that is attached to the bug-reports in Bugzilla also contains, for each voter, a link to a voter-page that lists all the bugs the voter in question has voted for split according to the project with which Bugzilla associates the bug. As far as we can tell, this "project" field is a new data element in the bug-reports that was not yet displayed in the original bug-reports that constitute the CVS-set of bugs. That is why we do not have such project-data for most of the bugs. Nevertheless via the voter-pages we can determine the project for those bugs for which at least one vote was cast. Contrary to what one might assume just a fraction of the bugs that we associate with code in the Firefox-branch of the Mozilla CVS are associated with Firefox by Bugzilla as well. Partly, this may be due to misattribution from our

part, but mostly this seems to be a reflection of the fact that Firefox shares code with other applications and the fact that it is based on code that was developed earlier for other projects. In addition to the project affiliation of CVS bugs the voter-pages also give the project affiliation for bugs that did not make it into that set — that is, bugs that did get votes but where not mentions in the commit-comments we looked at. The union of all these bugs constitutes the voter-set of bugs. We did not retrieve the complete Firefox-set of bugs, but we do not which part of the voter-set overlaps with it and that another part of the CVS-set should do as well. Finally, our set of Firefox-voters is by looking at the voter-lists of all bugs in the voter-set that are associated with Firefox through Bugzilla and taking the union of the sets of voters listed in those lists. Table 1 gives a short summary of the manner in which we obtained our sets described above.

Table 1. Summary indication of source of the main sets of bugs and voters.

	Bugs	**Voters**
CVS	Bugs associated with Firefox code before version 2.0.	Union of lists of voters on bugs in set defined in previous column.
Votes	All bugs that received one or more votes by voters on the CVS bugs.	As above, ceteris paribus.
Firefox	All the bugs associated with Firefox in the listing of bugs per voter.	” ”

Another and, for now, final detail about the data preparation concerns the way we link voter-identities with other activities by the same people in the bug resolution process. For this we rely on the fact that up until recently the voters as well as participants to the bug forum discussions, bug-reporters, bug-assignees, etcetera, were identified by their email address. Hence we could assign various bug-activities to the voters by matching these email addresses. This method works fine for the bugs that we focus on for this study. However, studies concerned with the most recent bugs would not be able to apply this method as Bugzilla has moved to enhanced identity management and does no longer provide the full email address for voters.

4 Characterizing Voters

A first dimension along which voters can be distinguished from non-voters, reported in Table 2, is with respect to both groups' status in the community.

Table 2 is a contingency table comparing community status and voting activity of participants in the bug resolution process for the intersection of 418 bugs with the following properties: they appear in the CVS; they have attracted votes; and they are associated with Firefox. As participants, we consider people

Table 2. Number of voters by status for participants in the resolution of bugs which have received at least one vote, are mentioned in the CVS, and are associated with the Firefox project ($n = 2968$; $\chi^2 = 366$, $df = 3$, p-value $= 5.243e^{-79}$).

	Status				
	Unconfirmed	New	Both	Other	Total
Vote	184	25	64	1408	1681
No Vote	286	139	182	680	1287

who have either contributed at least one comment to at least one discussion thread or who have cast at least one vote for these bugs. Among the 418 bugs of interest, this definition yields a total of 2968 participants, 1681 of whom have cast a vote and 1755 of whom have written at least one message, which implies that 1213 participants have voted but never written a message. A first conclusion that can be drawn from these numbers, then, is that most voters do not engage in other activities. If we go one step further and check participation on a bug-by-bug basis, this finding becomes even more pronounced: typically, voters who are active in the bugs in the set tend to engage in this activity on bugs for which they did not vote.

In order to gauge the status of participants we rely on the CanConfirm privilege mentioned earlier. In particular, we check for each participant whether he or she has ever declared bugs contained in the set of 37408 bugs that appeared in the CVS logs. For those participants who did declare bugs we look at the initial status of those bugs. If all the bugs that a participant declared start with status NEW the participant is considered to belong to the core; if all the bugs start with status UNCONFIRMED the participant belongs to the periphery; if some bugs start with UNCONFIRMED and others with NEW the participant is considered to be a freshly joined or a freshly expelled member of core; finally if the participant has not declared any bug, he or she is classified as member of the outer-periphery, here denoted as "Other".

Given all this, the main conclusion from table 2 is that the people who cast a vote are mostly, yet not exclusively, outsiders. Interestingly, this finding also holds for the status of the people who voted for any one of the bugs that appeared in the CVS logs. Of the 11850 voters among these 3787 bugs 10269 have never declared a bug while only 283 participants have had all the bugs they declared accepted as NEW right-away. Furthermore, when we consider the top 1000 most active voters, 688 among them can still be classified as outsiders while only 36 belong to the core.

Figure 2 gives some indications about the voting activities of the people who have cast a vote for at least one of the 418 bugs in the latter sample. It shows the distribution of the number of bugs to which people have cast their vote as split according to projects and status. Actual outsiders, with status Other, tend to declare "pure" Firefox bugs while contributors with another status disperse their votes much more.

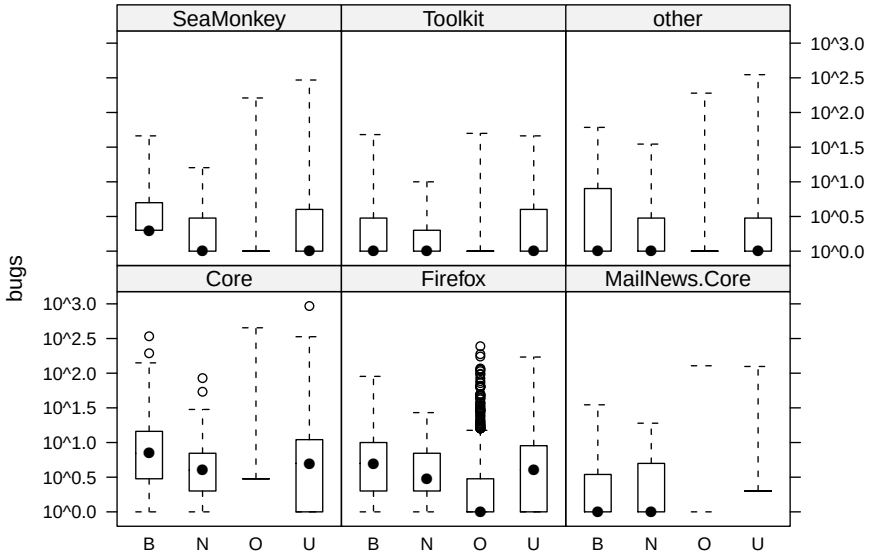

Fig. 2. Distribution of number of bug declared by status per project. B = both; N = new; O = other; U = unconfirmed.

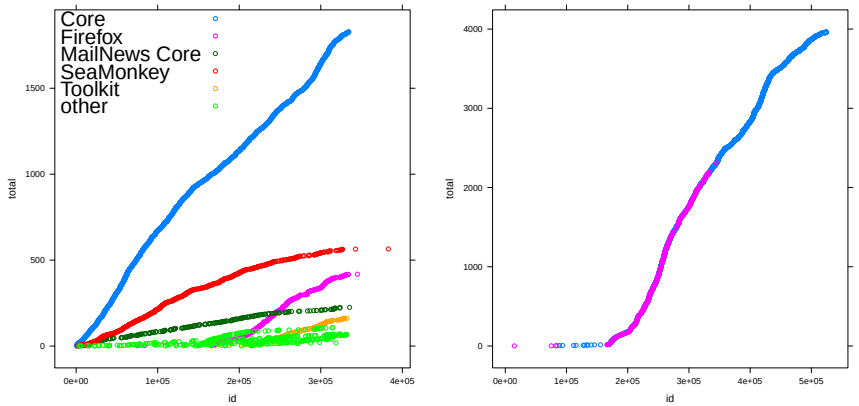

Fig. 3. Cumulative sum of bugs per project against bug-id; in the plot on the left specifically for Firefox (see text).

Figure 3 tries to shed some light on the timing of bug activity. Votes do not come with timestamps in the Bugzilla records, but we know that bug-ids are assigned sequentially. Figure 3 shows how many bugs had received votes before a given bug id against the sequence number that is used to identify them and this gives a rough indication about the distribution of voting over time. On the left is shown this distribution for bugs with votes and that appear in the CVS logs: typically, a steady proportion of bugs attract votes; not all projects are active at the same time; and some, for instance SeaMonkey, are gradually abandoned while others take off. The right graph focuses on bugs associated with the Firefox project more in particular. This graph includes the bugs that were voted on by the people who voted for the 418 Firefox CVS bugs that are represented in purple in the middle of the graph. The picture that appears is one of a very *loyal* electorate. Note that we are looking at all the votes that were cast by people who voted during the specific time window which is occupied by the 418 Firefox CVS bugs. Normally, one would expect a certain level of turnover among these people, implying that number of people who continue to vote would decrease over time. Nevertheless the ratio between increase in the number of bugs with votes and the increase in the total number of bugs looks more or less stable. So either people continue to vote, or abandonment of the project by some people is compensated by increased voting activity of the people who remain. In addition, the slope might also have been influenced by a change in the proportion of bug-ids that are attributed to the Firefox project.

5 Analyzing Bugs

Figure 4 shows the share of each project among the bugs in the CVS-set that have at attracted at least one vote. With 418 out of 3785 bugs Firefox is far from the most heavily represented project in this set. The most heavily represented project is "Core", which can be explained by the fact that "Core" concerns the foundation of code that is common to most Mozilla projects. The SeaMonkey project "is a community effort to develop the SeaMonkey all-in-one internet application suite" (`http://www.seamonkey-project.org/`). In a sense, it pursues a strategy that Firefox explicitly chose not to follow and it is a little ironic that there are so many SeaMonkey bugs mentioned in commits related to Firefox code. This may be a reflection of the fact that these siblings have a lot of code in common. "MailNews Core" is harder to explain as this project is concerned with code for email clients, which Firefox, in contrast to SeaMonkey, choose not to include. "Toolkit", finally, includes cross-platform components such as "Gecko" the rendering engine for web-pages — which is of course important to Firefox in particular.

In order to compare the bugs that attract votes with those that don't, Table 3 gives the estimations of a generalized linear model, assuming a Poisson distribution for the variable totalVotes, which is a numeric variable whose value represents the total number of votes that a bug has attracted. The estimates

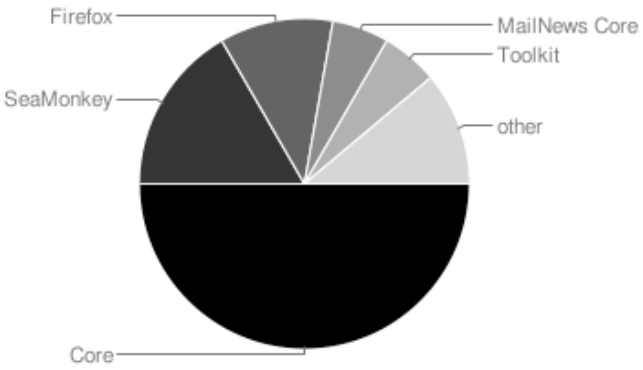

Fig. 4. Pie chart showing the share of Mozilla projects among the bugs that received bugs in our sample.

are based on data from a subset of about ten thousand cases, bugs for which we don't have complete information, very old bugs with a low bug-number and other outliers having been removed. Please refer to [3] for a detailed description of these variables in so far as they are not explained below.

The number of votes is statistically correlated with NumAttention, which relates to the number of people who have put themselves on the CC-list associated with the bug so that they can follow the progress on the bug resolution, i.e. with the attention received by a bug.

More generaly, more votes tend to be associated with "problems" in solving a bug and notably to neglect: patches, which counts the number of different patches needed to solve a bug; duplicatesBeforeLastResolved, which indicates repeated declarations of the same bug; Ltpsassign, the log of the time need for the bug to be assigned to someone; numberOfTimesAssigned, the number of times a bug has been assigned; or else bugWasReopened, a self-explanatory dummy variable, are all variables of this kind. Conversely, high values for com_com and com_authrate, which are indicative for a high level of activity and a lot of commitment on a bug, are correlated with a lower number of votes: votes would be less needed when there is enough commitment on a given bug.

Interestingly then, several variables generally relevant for the patching and triage of bugs are not or only weakly significant here: numberOfEditsByBugReporter and numberOfEditsByLastAssignee, two variables that reflect the level of activity of the people who are most directly involved with the resolution of a bug; severity, which reflects the current community estimate of the severity of a bug; priority, the assessment by the community of the importance of a bug; and dependson, the number of other bugs which have been found to depend on a given bug.

Finally, and closer to our interest in this article, I/We, which represents the ratio between the number of times that the personal pronouns I and We appear in the bug thread, is highly significant: bugs patched in "I-mode" tend to

Table 3. Generalized linear model with Poisson distribution for dependent variable totalVotes based on 10763 observations.

Parameter	Est.	Std.Er	[Wald	95%]	χ^2	Pr > χ^2
Intercept	0.3231	0.1697	-0.0096	0.6558	3.62	0.057
numberOfEditsByBugReporter	0.0061	0.0029	0.0005	0.0118	4.57	0.0326
numberOfEditsByLastAssignee	-0.0025	0.0014	-0.0052	0.0003	3.13	0.0771
NoSgRestriction	0.5972	0.1046	0.3922	0.8022	32.6	<.0001
numberOfTimesAssigned	0.1064	0.006	0.0946	0.1182	311.23	<.0001
bugWasReopened	0.3264	0.0264	0.3781	0.2747	153.14	<.0001
nauth	0.1799	0.021	0.1387	0.2212	73.19	<.0001
patches	0.1217	0.0032	0.1154	0.1281	1410.15	<.0001
attach_patch	0.1065	0.0049	0.097	0.116	481.08	<.0001
com_com	-0.5699	0.0152	-0.5997	-0.54	1400.64	<.0001
comauthrate	-0.2283	0.0154	-0.2584	-0.1981	220.65	<.0001
severity	0.0341	0.0131	0.0085	0.0597	6.82	0.009
priority	0.0053	0.0097	-0.0138	0.0244	0.3	0.5848
priorityNotIncreased	-0.133	0.0698	-0.2698	0.0039	3.63	0.0568
priorityNotDecreased	0.1907	0.0461	0.1005	0.281	17.15	<.0001
severityNotIncreased	-0.321	0.0295	-0.3788	-0.2632	118.44	<.0001
severityNotDecreased	-0.4144	0.0403	-0.4934	-0.3354	105.71	<.0001
nfile	-0.0068	0.0009	-0.0086	-0.0051	58.35	<.0001
dependson	0.0216	0.0075	0.007	0.0363	8.39	0.0038
blocked	0.0561	0.0037	0.0489	0.0633	232.89	<.0001
NumAttention	1.0415	0.0164	1.0093	1.0737	4018.84	<.0001
version2 1.7	1.325	0.1105	1.1084	1.5416	143.73	<.0001
version2 Trunk	-0.4801	0.0285	-0.536	-0.4242	283.24	<.0001
Ltpsassign	0.0822	0.0053	0.0718	0.0926	239.27	<.0001
DuplicatesBeforeLastResolved	0.1741	0.0044	0.1654	0.1828	1531.72	<.0001
UX	0.3312	0.0454	0.2421	0.4202	53.12	<.0001
I/we	0.0537	0.0014	0.0509	0.0565	1383.95	<.0001
os3 NonWin	0.1279	0.0299	0.0693	0.1864	18.31	<.0001
OC	0.1415	0.0278	0.0871	0.196	25.94	<.0001
CO	0.424	0.0361	0.3533	0.4947	138.19	<.0001
OO	0.8045	0.0366	0.7327	0.8763	482.33	<.0001
Scale	1	0	1	1		

recieve more votes, either because people vote for their own bugs or because bug patching in I-mode is associated with the involvement of peripheral members of the community. The significance of OC, CO, and esspecially of OO — dummy variables indicating that the bug reporter and the first "patcher" for that bug stem from periphery and core (in case of OC), core and periphery (CO), or both from the periphery (OO), respectively — tends to support the view that the involvement of the periphery in patching a bug would be statistically associated with more votes, even while controlling by attention, commitment, neglect, and various other problems affecting bug patching.

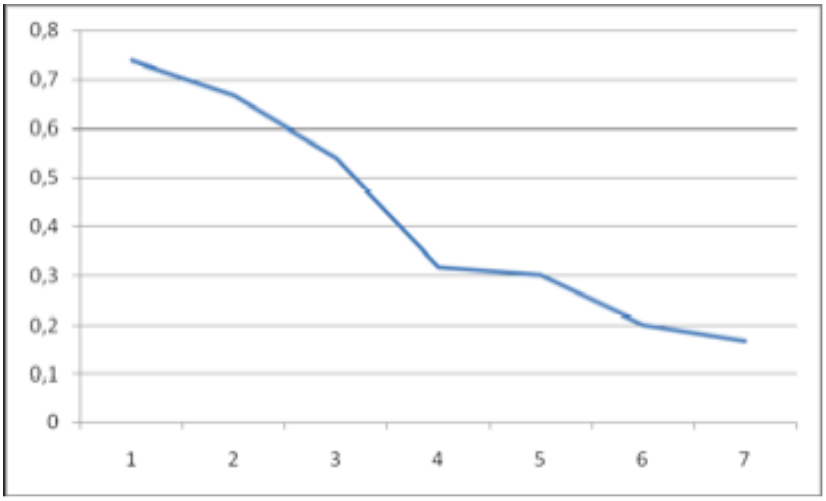

Fig. 5. Proportion of bugs that received patches from the periphery only among all bugs whose first patch was proposed by a member of the periphery (y-axis) against number of votes as average over a 5-vote range (i.e. $1 = 0$ votes, $2 = 1 - 5$ votes, $3 = 6 - 10$ votes etc.) on the x-axis.

In this last respect, Figure 5 is the result of an attempt to delve deeper into the relationship between patching and voting in the context of the relationship between periphery and core. Most of the patches are proposed by people from the core of the Firefox community. In many cases the first patch that is proposed is accepted as the solution for a bug. There are however a few cases in which multiple patches are proposed before a final patch is accepted as the solution for a bug. Figure 5 focuses on bugs whose first patch was proposed by a member of the periphery. We compute how many of these bugs do not receive any subsequent patches by members from the core, relative to bugs that have received zero votes, one to five votes, six-to-ten votes, and so further, respectively. What Figure 5 shows is that there is a clear increase in the level of participation of core members of the community when the number of votes increases. A possible interpretation of this could be that votes are used by the periphery to attract the attention of the core.

6 Conclusion

When Blake Ross and Dave Hyatt initiated Firefox, they established themselves as benevolent dictators fulfilling the *volonté générale*, to borrow from Rousseau, while emphatically reserving the right to ignore the *volonté de tous*. However, and contrary to acounts of voting in open source communities which tend to present it as a method to arrive at a fair representation of the will of *the* community, they allowed voting as as a channel through which *voices from outside the*

community could be heard. Consequently, we found that most votes originate in the outer periphery. Bugs that attract most votes tend to be bugs that are relatively neglected or bugs where the periphery is heavily involved. Hence one can surmise that the votes were cast in order to attract the attention from the core. It would be interesting to see whether there are additional mechanisms apart from the votes that help the core to focus its attention.

References

1. Alan Cox. Cathedrals, bazaars and the town council. Available at http://www.linux.org.uk/Papers_CathPaper.cs, 1998.
2. Jean-Michel Dalle and Matthijs 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.
3. Jean-Michel Dalle, Matthijs den Besten, and Héla Masmoudi. Channelling Firefox developers: Mom and dad aren't happy yet. In *Proceedings of the Fourth International Conference on Open Source Systems*, Milan, September 2008.
4. Andre L. Delbecq and Andrew H. Van de Ven. A group process model for problem identification and program planning. *Journal of Applied Behavioral Science*, 7(4):466–492, July 1971.
5. Albert O. Hirschman. *Exit, voice, and loyalty.* Harvard University Press, 1970.
6. Jessica Livingston. Blake Ross; creator, Firefox. In *Founders at Work: Stories of Startups' Early Days.* Apress, 2007.
7. Alan MacCormack, John Rusnak, and Carliss Y. Baldwin. Exploring the structure of complex software designs: An empirical study of open source and proprietary code. *Management Science*, 52(7):1015–1030, July 2006.
8. Héla Masmoudi, Matthijs den Besten, Claude de Loupy, and Jean-Michel Dalle. Peeling the onion: The words and actions that distinguish core from periphery in Firefox bug reports, and how they interact together. In Keven Crowston and Cornelia Boldyreff, editors, *Proceedings of the Fifth International Conference on Open Source Systems*, 2009.
9. Juan Mateos Garcia and W. Edward Steinmueller. Applying the open source development model to knowledge work. INK Open Source Research Working Paper 2, SPRU - Science and Technolgy Policy Research, University of Sussex, UK, January 2003.
10. Audris Mockus, Roy T. Fielding, and James D. Herbsleb. Two case studies of open source software development: Apache and mozilla. *ACM Trans. Softw. Eng. Methodol.*, 11(3):309–346, 2002.
11. Siobhán O'Mahony. The governance of open source initiatives: what does it mean to be community managed? *Journal of Management and Governance*, 11(2):139–150, May 2007.
12. Siobhán O'Mahony and Fabrizio Ferraro. The emergence of governance in an open source community. *Academy of Management Journal*, 50(5):1079–1106, 2007.
13. Gabriel Ripoche. *Sur les traces de Bugzilla.* PhD thesis, Université Paris XI, 2006.
14. Diederik W. van Liere. How shallow is a bug? Technical report, Rotmon School of Management, University of Toronto, November 16 2009.