“Peeling the Onion”: Analyzing Core and Periphery in the Firefox Community with Text-mining Methods

Héla Masmoudi[1], Quentin Peigné[1], Claude de Loupy[3,4], Matthijs den Besten[2] and Jean-Michel Dalle[1,*]

1. Université Pierre et Marie Curie, Paris, France.
2. Ecole Polytechnique, Paris France
3. Syllabs, Paris, France
4. University of Paris 10, MoDyCo Laboratory, Paris, France
** Corresponding author*

Abstract

According to the largely accepted “onion-model” of the organization of open source software development, an open source project relies on a core of developers assisted by a larger periphery of users. Following this characterization of the division of labor between a core and a periphery, and with the help of text-mining methods, we study the treatment of bugs in the Firefox community through the discussions and actions recorded in Mozilla’s issue tracking system Bugzilla. We mostly focus on the interactions between core and periphery, and suggest that these processes appear more diverse and subtle than initially thought, including late and/or stigmergic entry by members of the periphery, while they also generally appear to be affected by a “sense of community” exhibited by members of the core.

Keywords

Open-Source, FLOSS, firefox, bugs, text-mining, core, periphery, onion model

1. Introduction and brief survey of the literature

Free/Libre Open Source Software (FLOSS) development is an open process. Nevertheless, this openness does not imply that associated processes are democratic or that effort is shared evenly among the community. On the contrary, Crowston and Howison (2006) posit that “projects are mostly quite hierarchical” based on a social network analysis of bug-fixing interactions within a large variety of open source projects. In a study of the Linux Kernel, Lee and Cole (2003) find that decision power resides with a small core of developers while less critical tasks are typically delegated to a larger supporting periphery, while a study of Apache and Mozilla leads Mockus et al. (2002) to suggest that typically “a group larger by an order of magnitude than the core will repair defects, and a yet larger group (by another order of magnitude) will report problems.” In an attempt to come up with a general description of the social organization of FLOSS projects, Crowston and Howison (2005) formulate the “onion model” of software development. According to this model a small group of core developers is surrounded by several layers of peripheral helpers ranging from occasional problem solvers close to the core to mainstream users whose contribution is limited to the occasional submission of crash reports. Rullani (2009) makes a valiant attempt to figure out what the periphery’s role could be and reconcile the “passivist” view of Raymond (1998) that the main contribution of the periphery is to be the collection of “eyeballs” that will make that “all bugs are shallow” with the more “activist” view of Lakhani (2006) who argues that peripheral “interference” is crucial for the health of an open source project. This latter view is also supported by Von Krogh et al. (2003) who further

focus on the “joining scripts” i.e. the typical activities and sequences of activities in which developers engage when they move from periphery to core.

We believe that this literature calls for deeper empirical investigations about the division of labor and about the interaction between core and periphery, in order to enhance our understanding of FLOSS and related online communities. In this respect, we explore in this paper the role of the core and the periphery in the case of the community associated with Mozilla’s Firefox Internet browser. We analyze activities related to Firefox of participants in Mozilla’s Bugzilla bug tracking system. With the help of computational linguistics tools, we expose patterns of interaction among people whose core or periphery status we identify on the basis of a technical “privilege” internally managed by the community. We find several pieces of evidence that suggest more diverse and more subtle patterns of interaction between core and periphery.

Section 2 describes the methodology used to treat bug-tracking data. Section 3 provides a broad characterization of the core and peripheral activities based on bug threads. Section 4 then presents a finer-grained analysis of core and peripheral actions using what we suggest to call “praxic alphabets”, which we use to identify sequences of actions specific to the core or to the periphery, as well as interactions between core and periphery. Section 5 concludes.

2. Selection and preparation of the bug report database

Previous research has identified bug reports as a primary way through which the community communicates with developers (Mockus et al., 2002). Furthermore, assuming that the organization of the community is reflected in the tools with which it coordinates its activities (Lanzara and Morner, 2005), traces of this structure should be visible in the community's bug tracking system. In the case of Firefox, the issue tracker is the Bugzilla bug tracking system hosted by the Mozilla foundation. There are however two complicating issues that we need to control for. First, the bug tracking system at Mozilla also maintains information that relates to other, possibly unrelated, projects also hosted by Mozilla and, second, given that anyone can submit bug reports and that Firefox is well known, the whole corpus of bug reports is likely to contain a lot of noise – noise that is filtered out at some level and never reaches most developers. To deal with both of these issues, we focus only on the subset of bug-reports that have had an effect on the code base: we focus on bugs whose numbers are identified in comments to revisions to code belonging to the Firefox branch, or Firebird or Phoenix branch (as Firefox was formally known) in a version of Mozilla's CVS code archive dating from 2007. Assuming that this heuristic procedure of identification is correct (but see Ayari et al. 2009), we expect that the bugs in our corpus cover bugs whose fixing contributed to the further development of Firefox. in the decade before 2007, our main corpus still contains information on over 37 000 bugs.

Bugzilla then marks new bug-reports differently depending on the status of the bug-

reporter. By default, a new bug-report is marked as “UNCONFIRMED”, while only a limited number of people who possess the “CanConfirm” privilege have their bug-reports immediately accepted as “NEW”. Others have to wait until one of these privileged people has vetted the “UNCONFIRMED” bug-report and confirmed it as properly “NEW”. Hence, we use the “CanConfirm” privilege as a proxy for the status of bug-reporters and other participants in the bug resolution process.[1] Based on the email addresses with which participants identify themselves in Bugzilla, we thus consider as member of the *core* a) contributors who have at least one bug report reported by them marked as “NEW” and b) for all bugs whose number is higher to the number of the first bug that they have reported as “NEW”. Conversely, we consider as member of the *periphery* a) all contributors who have reported “UNCONFIRMED” bugs until the number of the bug that they first report as “NEW”, when it exists and b) all contributors who have never reported a “NEW” bug.

Bugzilla then traces a rich set of information allowing us to trace the resolution of bugs. In particular, each bug-report generates a discussion thread and all the messages exchanged between participants in the bug resolution are recorded. In addition, there are a number of metadata fields indicating, for instance, the perceived severity and priority of a bug, which subsystem the bug relates to, or else to whom, if anyone, the bug is assigned. There is also a list of attachments –proposed patches and contextual elements such as screenshots –for each bug, and

[1] Further research could compare the assignment according to a CanConfirm privilege to an assignment derived from social network analysis (SNA), similar to the comparison between SNA and developer lists carried out by Crowston et al. (2006).

a complete log of all changes that are made to metadata fields in the bug report.

To analyze this rich dataset, we combined two sets of methods inspired by computational linguistics. Indeed, as far as we know, apart from the interesting work of (Ripoche and Sansonnet, 2006), text-mining techniques have not really been used on such data archives. The first set of methods, described in more detail in Section 3, focuses on messages exchanged in the bug thread. By statistically looking at the words that people use, we discern differences in discourse and representation between core and periphery. Our second set of methods, described more detail in Section 4, is, we think, an original way to include in the analysis the many different actions of contributors related to a bug report, including all actions associated with metadata. Focusing on a subset of actions, we encode each action in a predefined “praxic alphabet” and analyze the strings of letters that this process yields.

3. Characterizing core and periphery with bug threads

A. Frequency of bug threads

Using the global corpus described above, we find that a proportion of 20-25% of the bugs are initiated by outsiders. Of the 6197 distinct email addresses that are associated with the opening of one or more of the bug reports in our corpus, 1713 are marked as core and 5118 as periphery while 634 switch from one to the other.

We also assign to the periphery all participants who never submit a bug report, i.e. 12219 email addresses, of which 6821 acted only once ever. Most of the peripheral contributors report only one bug (3851); 620 report two bugs and 386 report more than two. The numbers of bugs reported by core developers is more evenly distributed and averages about 16 reports. Considering all actions taken and comments contributed to the bugs reports, about 85% can be traced back to members of the core. Furthermore, lengthier interactions tend to be associated with a lower involvement from the core (N): see Figure 1.

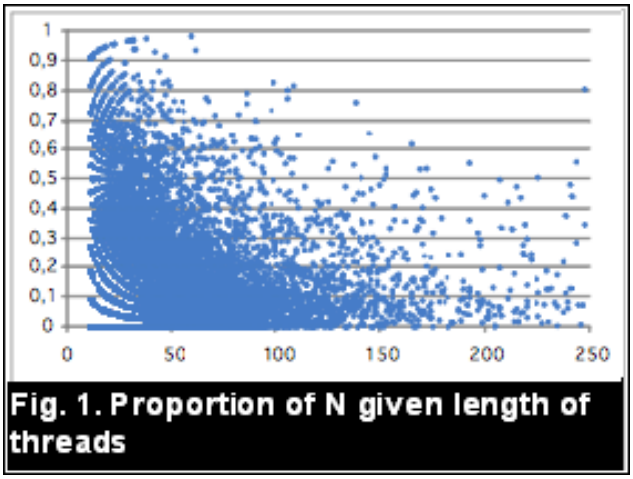

Fig. 1. Proportion of N given length of threads

We then plot the frequency of threads with given proportions of core vs. Peripheral involvement, measured by the percentage of actions in the bug thread from core and peripheral members of the community. Figure 2a shows that the frequency of threads decreases linearly in log scale with the proportion of peripheral actions for a

large part of the spectrum. That is to say, exponentially more problems are solved with a linear increase in the proportion of core developers.

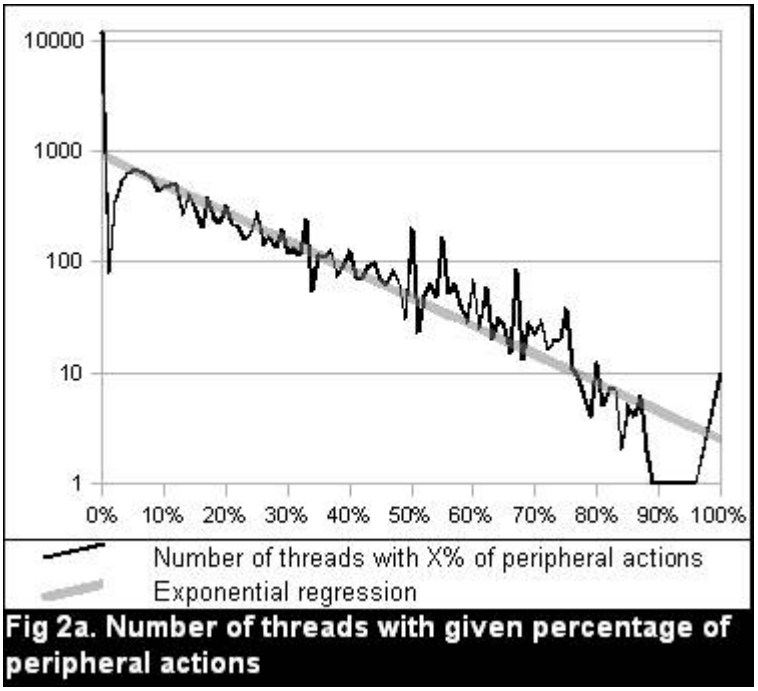

Fig 2a. Number of threads with given percentage of peripheral actions

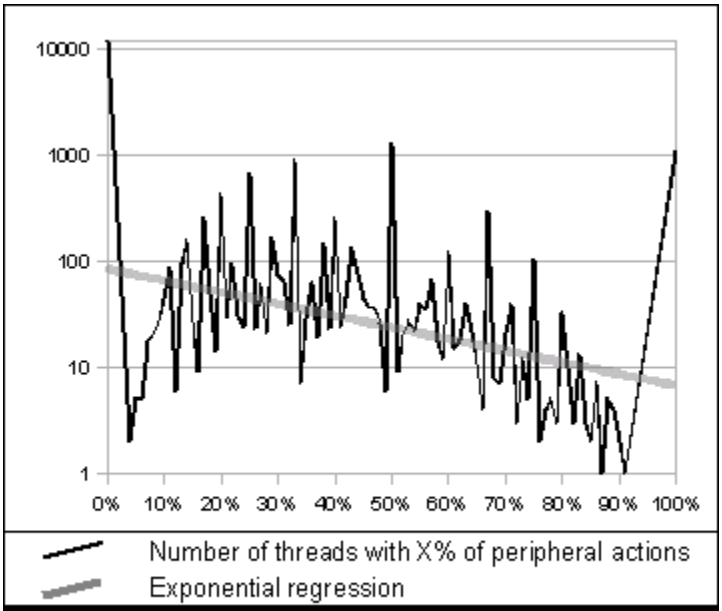

Fig 2b. Number of threads with given percentage of peripheral a-actions

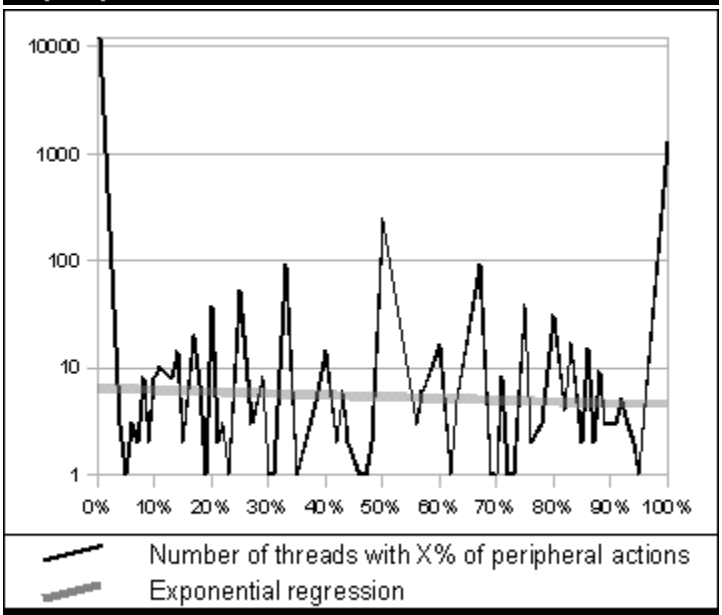

Fig 2c. Number of threads with given percentage of peripheral p-actions

We interpret this finding as suggesting an increased frequency of group discussions between core members i.e. between individuals who “know” each other or at least know that they share similar privileges, which we could characterize as a “wedding table” effect, discussions being typically more frequent at tables where more people knew each other ex ante compared to discussions at tables with “outsiders”. This interpretation is supported by the fact that a similar pattern holds when restricting the analysis to messages only, while it does *not* hold for other kind of actions such as the submission of patches or addition of an email address to the CC list of a bug, as reported by Figures 2b and 2c respectively.

B. Entry and words used

We now turn to the textual content analysis of bug threads. Among the various tools available to perform this kind of analysis, we have selected Lexico[2], which is particularly helpful in estimating the likelihood of occurrences of words and other items and comparing these frequency-estimates among different parts of the corpus (Lamalle *et al.* 2003). A crucial metric in this type of analysis is called *specificity*. This metric is an indicator of how specific certain terms are to the parts of the corpus in which they occur. The sign of the metric indicates whether terms are over- or under-employed in specific parts (Lebart and Salem 1994).

[2] http://www.cavi.univ-paris3.fr/Ilpga/ilpga/tal/lexicoWWW/index.htm

Because of computational limits, we specifically look at two subsets of the bug-reports: the first, “cvs-sub”, consists of 2000 bugs with bug id between 54452 and 73095[3]; the second, “cvs-mile”, consists of the 694 bugs associated with a target milestone specifying a version of Firefox (or its previous incarnations Firebird and Phoenix).

Reports in both of these corpus were tagged and subsequently partitioned with the help of tags identifying:

1) whether the event recorded was originated from the core or the periphery;
2) each contributor who participated in the bug resolution in order of appearance from the first to the 25th (letters a-y) and subsequent contributors (letter z);

A first result, related to Figure 1 and presented in Fig. 3, concerns the identity of people involved in the discussion. People who enter later in the discussion are less likely to come from the core than earlier on.

[3] Note that these are 2000 consecutive bugs that were traced in the comments of the CVS log; the fact that the difference in bug id leaves space for about 20000 bugs suggests that 90% of bug reports on Mozilla’s bugzilla are either resolved without affecting the code base or deal with code that is not related to Firefox.

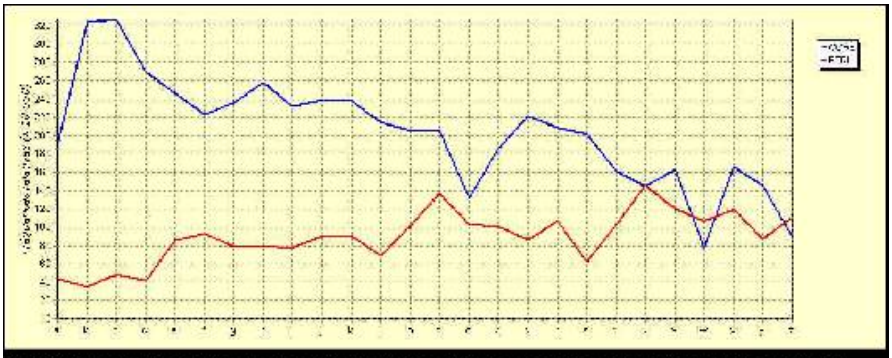

Fig 3. Frequency that an action stems from a core or periphery member for the n-th (coded by alphabetical letters) different actor in the thread

Tables 1 and 2 then present words most specific to comments from the core and the periphery, respectively. In particular, the relative importance of words like "*We*" and "*I*" can be interpreted, again, as being indicative of a sense of community among core contributors compared to a more atomistic periphery (Rullani, 2009). There is also a clear distinction between core members dealing with technical issues and using corresponding terms, on the one hand, and peripheral members who approach the black box of Firefox from the outside. Noteworthy as well is the relative importance of "Windows" and "NT" among the periphery. The words "#CATTACH" and "#DUPLICATE" in the tables are short hand for the standard expressions "Created an attachment" and "This bug has been marked as a duplicate", respectively. Core members would seem to be more involved in these activities too.[4]

[4] However, other investigations based on a slightly stricter definition of core membership in which people who had recently obtained the CanConfirm privilege were still marked as

Table 1. Words that are most specific for core and periphery members in corpus "cvs-sub"

Core				Periphery		
	Freq.	Specif.			Freq.	Specif.
we	6964	***		Mozilla	1131	***
fix	3402	***		reply	211	***
line	3368	***		signature	193	***
int	1598	***		Results	189	***
const	1353	***		dbg	156	***
bytes	1317	***		quoted	138	***
unsigned	1022	***		sig	128	***
cpp	2308	50		text	644	46
patch	5434	48		page	662	44
seamonkey	516	39		browser	455	44
checked	1503	38		checking	189	34
JS	806	37		IE	206	32
Marking	559	37		download	236	29
branch	1063	36		application	185	28
#CATTACH	3689	35		color	175	28
builds	1064	34		mail	413	25
trunk	1349	33		top	216	25
review	800	32		message	429	24
code	2571	31		comment	393	24
changes	1158	25		dist	172	23
js	1245	24		email	166	21
looks	859	24		URL	288	20
verified	557	24		XML	157	20
We	1161	19		user	499	19
profile	812	18		behavior	199	18
crash	810	18		I	5574	17
need	2149	17		HTML	235	17
string	721	17		Windows	270	16

periphery we actually found the identification of duplicates to be a significant activity for periphery members, which suggests that this marking of duplicates might also be part of a "joining script" (Masmoudi et al., 2009).

Table 2. Words that are most specific for core and periphery members in corpus "cvs-mile"

Core				Periphery		
	Freq	Spec			Freq	Spec
mass	213	***		Windows	1573	***
xul	368	***		Mozilla	1273	***
cvsroot	389	***		Gecko	887	***
Checking	422	***		rv	858	***
toolkit	470	***		Firefox	1459	47
done	639	***		NT	556	43
content	766	***		Results	322	38
revision	961	***		Agent	234	36
mozilla	1223	***		User	253	33
we	1887	***		Build	240	33
previous	537	47		Identifier	216	32
branch	596	46		Reproducible	223	31
trunk	713	46		Always	221	31
base	288	43		Reproduce	222	30
#CATTACH	1392	39		i	395	22
browser	1317	37		US	752	20
#DUPLICATE	1154	36		Steps	233	20
We	377	35		problem	669	19
Help	391	35		click	487	16
components	343	34		option	232	16
chrome	331	33		I	5451	15
patch	1845	33		Firebird	345	15
help	499	25		bookmark	302	15
cpp	240	23		bar	483	14
changes	415	22		am	199	14
checked	452	22		In	941	13
need	672	22		reply	709	13

4. Characterizing core and periphery with praxic alphabets

In this section, we look at how actions relate to each other. In order to do so, we first define a subset of actions of particular interest and encode these actions in an alphabet. We then code bug reports and their resolution paths into character strings made up of this praxic alphabet. Having thus transformed the corpus, we are able to apply standard text analysis techniques on the data. We focus here to the frequency of the occurrence of actions and pairs of actions – to what in computational linguistics is known as unigrams and bi-grams.

We consider seven types of events or actions in our analysis:

1) C – the creation of an attachment other than a patch, such as a screenshot;
2) D – the identification of a duplicate of a bug;
3) G – the assignment of a person to take the lead in the bug resolution process;
4) P – the creation of a patch, i.e. a suggested change in the code base;
5) R – a change in the priority assigned to the bug;
6) V – a change in the severity assigned to the bug;
7) X – the declaration of the bug as resolved.

These seven types have been chosen as characteristic of actions in bug resolution that seem crucial: C and D correspond to the provision of *contextual* information that helps understand better the nature of the bug after it has been reported (Dalle et al. 2008); G, R, and V relate to the process of *triage* – a process in which the bug is

assessed and matched to the available resources within the community (Villa, 2003); P and X concern bug-*resolution* in itself.

Using this alphabetic coding, Tables 3 and 4 present the importance of a given action among core or peripheral users and the proportion of core and periphery for a given action, respectively. Actions related to management (e.g. nominating assignees, triaging bugs by their priority level, and declaring bugs as solved) are predominantly realized by core members of the community, while the periphery appears responsible for the provision of contextual elements and for the identification of duplicates.

Table 3. Frequencies of action unigrams by origin (core or periphery)

Action-Core	Proportion		Action-Periphery	Proportion
C	17,3%		c	24,8%
D	12,7%		d	24,5%
G	11,3%		g	5,0%
P	36,2%		p	38,5%
R	1,1%		r	0,3%
V	1,6%		v	1,6%
X	19,8%		x	5,3%

Table 4. Frequencies of actions and core/periphery proportion per action

Action	Global freq.	Core part	Peri part
C	18,0%	86,7%	13,3%
D	13,8%	83,0%	17,0%
G	10,7%	95,5%	4,5%
P	36,4%	89,8%	10,2%
R	1,0%	97,1%	2,9%
V	1,6%	90,6%	9,4%
X	18,4%	97,2%	2,8%

Table 5 now shows the probability for a given action (in rows) to be followed by another action (in columns). Each of these probabilities is compared to what would have been the expected frequency of the next action had the occurrence of both actions been independent events. Probabilities that deviate from at least 20% of this norm are reported in blue if above the norm and in light gray if below. The high values of probabilities in the diagonal suggest that a repetition of actions is likely during the bug resolution process. Only X deviates from this pattern, which seems reasonable considering that some other actions are probably necessary before a bug that has been declared “resolved” could be declared “resolved” again. Nor is “G” extremely likely to repeat itself, just slightly above average, from which we can infer that people who have been assigned a bug rarely pass the bug, or at least not immediately. Last, the exceptionally high probability of D to followed by another D suggests some kind of cascading effect in which the discovery of one duplicate leads to the discovery of even more duplicates. Another compatible explanation for this sequence, however, is that this is the only kind of activity that is being carried out after a bug has been resolved. The high probability of X to be followed by D reinforces the latter interpretation.

Table 5. Frequencies of actions after given action compared to theoretical levels +/- 20%							
	C	D	G	P	R	V	X
Theory	18,0%	13,8%	10,7%	36,4%	1,0%	1,6%	18,4%
C	26,7%	8,0%	9,8%	36,2%	1,7%	1,6%	16,0%
D	9,1%	59,2%	6,2%	14,8%	0,7%	1,4%	8,5%
G	19,8%	7,4%	10,8%	35,8%	1,4%	1,7%	23,2%
P	11,5%	4,2%	11,0%	42,3%	0,7%	0,5%	29,8%
R	22,5%	8,5%	9,5%	37,5%	2,4%	1,4%	18,2%
V	19,9%	13,3%	18,8%	29,0%	1,1%	6,3%	11,6%
X	19,1%	39,8%	5,3%	23,6%	0,7%	1,8%	9,7%
blue : greater than theory +20% gray : lesser than theory -20%							

Another phenomenon worth interpreting has to do with the higher probability of RC, VC and VG couples. RC and VC indicate that there is a feedback loop from triage to provision of context. That is, changes of severity or priority tend to be met with and demand for and provision of attachments to illustrate the problem. The high probability of VG on the other hand is likely to be a reflection of the successful completion of the process of triaging at the end of which the person who is most likely to be able to resolve the bug is assigned to carry out that task.

The interactions presented above do not take the originator of the actions into account. In order to do so, we adjust our alphabet and transform into *lower case* all praxic letters (actions) that originate from a member of the periphery while we leave the actions originating from a core member in upper case. In a similar vein to Table 5, Table 6 presents the matrix of interactions with this enhanced alphabet. Strikingly, actions from the core are mostly followed by other actions from the core and the sequences of actions that are over-represented in comparison to their expected

level follow closely the general sequences of actions described above. Similarly, but no less strikingly, actions from the periphery are generally followed by other actions from the periphery, while they are also globally follow the patterns established in Table 5.

Table 5. Frequencies of actions after given Core or Peripheral actions compared to theoretical level +/- 10%

	C	D	G	P	R	V	X	c	d	g	p	r	v	x
Exp.	17,3%	12,7%	11,3%	36,2%	1,1%	1,6%	19,8%	24,8%	24,5%	5,0%	38,5%	0,3%	1,6%	5,3%
C	**23,4%**	6,6%	8,7%	37,1%	**1,8%**	1,4%	17,2%	1,2%	1,0%	0,2%	1,2%	0,0%	0,0%	0,2%
D	7,6%	**50,0%**	5,8%	13,6%	0,7%	1,4%	8,9%	1,7%	8,1%	0,2%	1,7%	0,0%	0,0%	0,2%
G	18,5%	6,5%	10,2%	32,9%	**1,3%**	1,5%	**22,7%**	1,5%	0,9%	0,5%	2,6%	0,1%	0,1%	0,6%
P	11,6%	3,6%	10,6%	**40,6%**	0,7%	0,5%	**31,0%**	0,3%	0,5%	0,1%	0,4%	0,0%	0,0%	0,2%
R	**21,3%**	7,6%	9,1%	36,0%	**2,4%**	1,2%	18,4%	0,8%	1,0%	0,1%	1,8%	0,0%	0,1%	0,0%
V	18,0%	11,5%	**17,9%**	26,9%	1,0%	**5,4%**	11,7%	2,0%	1,8%	0,4%	2,6%	0,1%	0,5%	0,2%
X	16,9%	**35,3%**	4,5%	21,5%	0,7%	1,7%	9,1%	2,1%	4,9%	0,6%	2,1%	0,0%	0,1%	0,5%
c	11,8%	7,5%	**13,8%**	12,2%	0,6%	**1,8%**	5,7%	**28,9%**	3,3%	1,4%	10,6%	0,2%	0,6%	1,8%
d	5,1%	**35,9%**	5,6%	8,1%	0,6%	0,9%	4,8%	3,3%	**28,7%**	1,5%	4,2%	0,0%	0,4%	1,0%
g	4,5%	3,0%	7,6%	6,1%	0,3%	1,0%	11,4%	11,4%	3,3%	3,7%	36,6%	0,0%	0,7%	**10,2%**
p	3,4%	3,7%	9,5%	6,0%	0,2%	0,4%	12,8%	5,3%	1,9%	**4,6%**	**47,1%**	0,1%	0,2%	4,8%
r	**30,2%**	4,7%	**18,6%**	14,0%	**2,3%**	**2,3%**	9,3%	2,3%	2,3%	0,0%	11,6%	0,0%	0,0%	2,3%
v	14,5%	7,9%	**21,6%**	14,5%	0,4%	**7,0%**	7,5%	4,0%	5,3%	3,1%	9,3%	**0,4%**	**3,1%**	1,3%
x	12,8%	13,7%	9,9%	10,9%	0,0%	**1,9%**	8,9%	7,0%	16,3%	0,3%	13,4%	0,0%	0,6%	4,2%

Exp : Expected level
Bold : At least 20% greater than expected
Gray : At least 20% less than expected

Table 6. Frequencies of actions after given Core or Peripheral actions compared to theoretical level +/- 10%

	C	D	G	P	R	V	X	c	d	g	p	r	v	x
Exp.	17,3%	12,7%	11,3%	36,2%	1,1%	1,6%	19,8%	24,8%	24,5%	5,0%	38,5%	0,3%	1,6%	5,3%
C	**23,4%**	6,6%	8,7%	**37,1%**	**1,8%**	1,4%	17,2%	1,2%	1,0%	0,2%	1,2%	0,0%	0,0%	0,2%
D	7,6%	**50,0%**	5,8%	13,6%	0,7%	1,4%	8,9%	1,7%	8,1%	0,2%	1,7%	0,0%	0,0%	0,2%
G	**18,5%**	6,5%	**10,2%**	**32,9%**	**1,3%**	**1,5%**	**22,7%**	1,5%	0,9%	0,5%	2,6%	0,1%	0,1%	0,6%
P	11,6%	3,6%	**10,6%**	**40,6%**	0,7%	0,5%	**31,0%**	0,3%	0,5%	0,1%	0,4%	0,0%	0,0%	0,2%
R	**21,3%**	7,6%	9,1%	**36,0%**	**2,4%**	1,2%	**18,4%**	0,8%	1,0%	0,1%	1,8%	0,0%	0,1%	0,0%
V	**18,0%**	**11,5%**	**17,9%**	26,9%	**1,0%**	**5,4%**	11,7%	2,0%	1,8%	0,4%	2,6%	0,1%	0,5%	0,2%
X	**16,9%**	**35,3%**	4,5%	21,5%	0,7%	**1,7%**	9,1%	2,1%	4,9%	0,6%	2,1%	0,0%	0,1%	0,5%
c	11,8%	7,5%	**13,8%**	12,2%	0,6%	**1,8%**	5,7%	**28,9%**	3,3%	1,4%	10,6%	0,2%	0,6%	1,8%
d	5,1%	**35,9%**	5,6%	8,1%	0,6%	0,9%	4,8%	3,3%	**28,7%**	1,5%	4,2%	0,0%	0,4%	1,0%
g	4,5%	3,0%	7,6%	6,1%	0,3%	1,0%	11,4%	11,4%	3,3%	3,7%	**36,6%**	0,0%	0,7%	**10,2%**
p	3,4%	3,7%	9,5%	6,0%	0,2%	0,4%	12,8%	5,3%	1,9%	**4,6%**	**47,1%**	0,1%	0,2%	**4,8%**
r	**30,2%**	4,7%	**18,6%**	14,0%	**2,3%**	**2,3%**	9,3%	2,3%	2,3%	0,0%	11,6%	0,0%	0,0%	2,3%
v	14,5%	7,9%	**21,6%**	14,5%	0,4%	**7,0%**	7,5%	4,0%	5,3%	3,1%	9,3%	**0,4%**	**3,1%**	1,3%
x	12,8%	**13,7%**	9,9%	10,9%	0,0%	**1,9%**	8,9%	7,0%	16,3%	0,3%	13,4%	0,0%	0,6%	4,2%

Exp : Expected level
Bold : At least 20% greater than expected
Gray : At least 20% less than expected

On the other hand, there are actions from the periphery that are more likely to trigger the attention from the core, most of which correspond however to bigrams (sequences of actions) that are generally frequent. However, a sequence like cG, the provision of contextual elements followed by the assignment of a person to the bug, was typically found by looking directly at bug threads as happening early, among the first actions in a thread, which is consistent with an “eyeballs” view of the periphery.

Compare now Table 5 with Table 7: while Table 5 shows the general probabilities of bigrams while Table 7 shows the *conditional* probabilities that actions follow each other when considering only the set of bigrams where the first action originated from a different "constituency" (core or periphery) than the second. The pattern revealed by this table roughly matches the general pattern of table 5. Vp interactions however stand out[5]. This Vp sequence is found in slightly over 50 bug report traces. A patch is submitted by a member of the periphery after severity was changed by a member of the core. We interpret this finding, based also on cursory analysis of bug threads, as suggesting that a change in severity can be a "signal" (be it a decrease or an increase since both are present in our corpus) which could "open the door" to the submission of a patch by the periphery. This might be linked both to an informal joining script, where members of the periphery would virtual dip their toes in the water by choosing to propose solutions to bugs where a severity signal has been sent[6], and/or to the stigmergic theory of coordination in online communities (Dalle & David, 2007) (Den Besten et al., 2008) which holds that coordination obtains as a consequence of signals being sent and followed by actions orienting the allocations of efforts within the community.

[5] And Rv, but there are only two occurrences of Rv in our data.

[6] We indeed found several occurrences where peripheral members who showed this type of behaviour were in the process of been given the CanConfirm privilege.

Table 7. Frequency of a particular action after given action from the other group (core / periphery) compared to theoretical level +/- 20%

	C	D	G	P	R	V	X	c	d	g	p	r	v	x
Exp.	19,3%	23,8%	20,2%	16,2%	1,4%	1,9%	17,2%	20,5%	39,9%	7,6%	20,9%	1,0%	2,2%	8,0%
C								**26,4%**	29,0%	**9,4%**	22,9%	**1,3%**	2,6%	8,4%
D								12,9%	**67,4%**	3,3%	11,6%	0,4%	1,1%	3,2%
G								23,9%	17,2%	**12,4%**	**29,4%**	**1,6%**	2,2%	**13,3%**
P								19,7%	31,7%	6,8%	**27,9%**	0,7%	2,0%	**11,3%**
R								22,4%	24,7%	**10,8%**	**27,8%**	**3,5%**	1,9%	8,9%
V								22,8%	23,4%	**10,6%**	**28,9%**	0,9%	**7,9%**	5,5%
X								19,8%	**49,5%**	5,4%	15,2%	0,4%	**3,1%**	6,7%
c	**27,3%**	12,8%	**25,2%**	19,3%	1,6%	**2,9%**	11,0%							
d	14,6%	**49,1%**	11,2%	12,7%	**1,4%**	1,4%	9,6%							
g	22,3%	7,9%	**28,8%**	16,8%	1,6%	1,0%	**21,5%**							
p	14,1%	9,4%	23,6%	**17,5%**	0,9%	1,0%	**33,6%**							
r	**29,6%**	9,2%	23,9%	16,9%	**4,9%**	2,1%	13,4%							
v	22,6%	10,2%	**27,8%**	18,7%	1,3%	**7,8%**	11,7%							
x	**26,5%**	27,0%	15,0%	13,9%	**1,7%**	2,1%	13,9%							

Exp : Expected level
Bold : At least 20% greater than expected
Gray : At least 20% less than expected

Conclusion

Focusing on the Firefox community, and using several methodologies derived from computational linguistics, we have suggested in this article that the interactions between the core and the periphery of an online community could be richer and more subtle and diverse than initially thought. In particular, and in addition to supplementary evidence of the existence of “joining scripts”, we have shown preliminary evidence of late entry of the periphery in the bug resolutions processes,

which could be connected to the identification of duplicates, of the possible existence of signals which could act as entry doors for the periphery, such as a modification of the severity of a bug by a member of the core, and generally of a sense of community exhibited by the core both in the form of its use of the pronoun “we/We”, compared to members of the periphery using the pronoun “I”, which is probably connected to what we have suggested to call a wedding table effect according to which the frequency of discussions between developers is severely affected by the status of these developers, discussions, and thus bug solving processes, being considerably more frequent with more members of the core. Far from drawing too strong conclusions from these insights on the Firefox community, we mostly believe that these approaches and methodologies offer a potential avenue for future empirical research on FLOSS and other online communities.

References

[Ayari et al., 2007] Ayari, K., Meshkinfam, P., Antoniol, G., and Penta, M. D. (2007). Threats on building models from CVS and Bugzilla repositories: the Mozilla case study. In *Proceedings of the 2007 conference of the center for advanced studies on Collaborative research*, pages 215–228, Richmond Hill, Ontario, Canada. ACM.

[Cox, 1998] Cox, A. (1998). Cathedrals, bazaars and the town council. Available at http://www.linux.org.uk/Papers_CathPaper.cs.

[Crowston et al., 2005] Crowston, K., Heckman, R., Annabi, H., and Masango, C. (2005). A structuration perspective on leadership in free/libre open source software teams. In *Proceedings of the first international conference on open source systems*, pages 9–15. Genoa, Italy.

[Crowston and Howison, 2005] Crowston, K. and Howison, J. (2005). The social structure of open source software development teams. *First Monday*, 10(2).

[Crowston and Howison, 2006] Crowston, K. and Howison, J. (2006). Hierarchy and centralization in free and open source software team communications. *Knowledge, Technology & Policy*, 18(4):65–85.

[Crowston et al., 2006] Crowston, K., Wei, K., Li, Q., and Howison, J. (2006). Core and periphery in free/libre open source software team communications. In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICCS-39)*.

[Dalle and David, 2007] Dalle, J.-M. and David, P. (2007). Simulating code growth in libre (open-source) mode. In Curien, N. and Brousseau, E., editors, *The Economics of the Internet*. Cambridge University Press, Cambridge, UK.

[Dalle et al., 2008] Dalle, J.-M., den Besten, M., and Masmoudi, H. (2008). Channelling Firefox developers: Mom and dad aren't happy yet. In *Proceedings of the Fourth International Conference on Open Source Systems*, Milan.

[den Besten et al., 2008] den Besten, M., Dalle, J.-M., and Galia, F. (2008). The

allocation of collaborative efforts in open-source software. *Information Economics and Policy*, 20(4):316–322.

[Lakhani, 2006] Lakhani, K. R. (2006). *The Core and the Periphery in Distributed and Self-Organizing Innovation Systems*. PhD thesis, MIT.

[Lanzara and Morner, 2005] Lanzara, G. F. and Morner, M. (2005). Artifacts rule! how organizing happens in open source software projects. In Czarniawska, B. and Hernes, T., editors, *Actor-Network Theory and Organizing*, pages 67–90. Copenhagen Business School Press.

[Lebart and Salem, 1994] Lebart, L. and Salem, A. (1994). *Statistique textuelle*.

[Lebart et al., 1998] Lebart, L., Salem, A., and Berry, L. (1998). *Exploring Textual Data*.

[Lee and Cole, 2003] Lee, G. K. and Cole, R. E. (2003). From a firm-based to a community-based model of knowledge creation: The case of the linux kernel development. *Organization Science*, 14(6):633–649.

[Masmoudi et al., 2009] Masmoudi, H., den Besten, M., de Loupy, C., and Dalle, J.-M. (2009). Peeling the onion: The words and actions that distinguish core from periphery in Firefox bug reports, and how they interact together. In Crowston, K. and Boldyreff, C., editors, *Proceedings of the Fifth International Conference on Open Source Systems*.

[Mockus et al., 2002] Mockus, A., Fielding, R. T., and Herbsleb, J. D. (2002). Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology*, 11(3):309–346.

[Moon and Sproull, 2000] Moon, J. Y. and Sproull, L. (2000). Essence of distributed work: The case of the Linux kernel. *First Monday*, 5(11).

[Raymond, 1998] Raymond, E. S. (1998). The cathedral and the bazaar. *First Monday*, 3.

[Ripoche and Sansonnet, 2006] Ripoche, G. and Sansonnet, J.-P. (2006). Experiences in automating the analysis of linguistic interactions for the study of distributed collectives. *Computer Supported Cooperative Work*, 15:149–183.

[Rullani, 2009] Rullani, F. (2009). The periphery on stage: Functions, properties and dynamic evolution of the periphery in the free/open source software community. Unpublished report.

[Villa, 2003] Villa, L. (2003). Large free software projects and bugzilla: Lessons from GNOME project QA. In *Proceedings of the Linux Symposium*, Ottawa, Canada.

[von Krogh et al., 2003] von Krogh, G., Spaeth, S., and Lakhani, K. R. (2003). Community, joining, and specialization in open source software innovation: a case study. *Research Policy*, 32:1217–1241.