On the Relationship between Newcomer Motivations and Contribution Barriers in Open Source Projects

Christoph Hannebauer **Volker Gruhn**

paluno – The Ruhr Institute for Software Technology
University of Duisburg-Essen, Germany
{first name.last name}@paluno.uni-due.de

ABSTRACT

There has been extensive research on the the factors that motivate software developers to contribute to an Open Source Software (OSS) project. Contribution barriers are the counterside to motivations and prevent newcomers from joining the OSS project. This study searches for relations between motivations and contribution barriers with a web-based survey of 117 developers who had recently contributed their first patch to either Mozilla or GNOME.

The results substantiate the hypothesis that newcomers' motivations mirror their mental models of the OSS project they are going to contribute to, and that the mental model determines the impact of contribution barriers. More generally, we propose a new model for the joining process to an OSS project that takes social properties, motivations, and contribution barriers into account.

ACM Classification Keywords

D.2.9 Software Engineering: Management—*programming teams*

Author Keywords

Open Source; Contribution Barriers; Newcomers; Survey

INTRODUCTION

A steady influx of new developers is crucial for the sustainability of an Open Source Software (OSS) project [4]. To gain new developers, the OSS project must not only attract them but also help them onboard the project. In the onboarding phase, the attracted developers may still fail to overcome the contribution barriers of the project and therefore do not join the OSS project's development team. Developers trying to contribute their first patch to a specific OSS project are called *newcomers* in the context of this OSS project [34], irrespective of their expertise with software development in general or with other OSS projects.

While there is some research about what contribution barriers exist [33], it is unclear which contribution barrier are more and which are less important. The importance of a contribution barrier is, on the one hand, a result of how often it affects newcomers. On the other hand, it is a consequence of the impact it has on each individual newcomer. While some contribution barriers are merely a nuisance, others might be the single reason why a newcomer refrains from contributing to an OSS project.

We conducted a survey with newcomers to the two OSS projects Mozilla and GNOME to find out which of the contribution barriers are important for newcomers. 117 newcomers answered to the web-based questionnaire. This data allowed a statistical analysis of a hypothesis regarding the relationship between contribution barriers and motivations derived in the next subsection:

Motivations and Contribution Barriers

According to the Theory of Cognitive Dissonance [9], a person feels discomfort if two of the person's opinions contradict each other. This is particularly the case when new information contradicts the expectation derived from existing information. This type of discomfort is called *dissonance* and its reduction is a natural stimulus just like hunger. In order to reduce dissonance, people may take a new point of view, but they may also just discard newly gained information if it does not fit their existing views.

Before newcomers decide that they want to contribute to an OSS project, they have a mental model of how an OSS project works and especially about the procedures and reception of their contribution. Like any model, this mental model is incomplete, although some newcomers may be aware of this incompleteness. Even if they are, they have unconscious or conscious assumptions about the procedures of their contribution, as they must have founded their decision to contribute on something. Thus, their motivation to contribute derives from this model. For example, if they assume that a contribution to an OSS project gains respect for the contributor, then they may derive the motivation to gain respect from the OSS community through their contribution. Accordingly, this can be used in the other direction: Contributors' motivations are indicators for their assumptions about the contribution before their contribution experience.

If the contribution experience contradicts the previous assumptions, dissonance arises. The mental model also includes assumptions about what parts of the contribution procedure are difficult and which are not. Hence, unexpected problems create

OpenSym '17, August 23–25, 2017, Galway, Ireland

© 2017 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ISBN 978-1-4503-5187-4/17/08. . . $15.00

DOI: https://doi.org/10.1145/3125433.3125446

more discomfort than expected problems. One way to reduce this dissonance is to refrain from the contribution: Either after accepting the new information about unexpected problems or through discarding existing information that served as foundation for the motivation, even if that information was in fact true. A newcomer may also ignore or accept the unexpected problems and continue with the contribution, if that is the lesser mental effort [9]. Since dissonance can be a reason to refrain from contribution, dissonance influences the perception of contribution barriers.

This application of the Theory of Cognitive Dissonance to OSS contributions leads to the following hypothesis:

> *Contribution barriers are less important for contributors whose motivations suggest that they expect these contribution barriers.*

Definition of Terms

Von Krogh et al. [38] first introduced the terms *contribution barrier* and *joining script* in the context of OSS projects. According to their definition, contribution barriers are hurdles that prevent newcomers to join an OSS project. *Joining scripts* are the processes that newcomers should follow in order to be granted commit access to the project's Version Control System (VCS). Typically, the joining script is not fully documented. Thus, newcomers need to learn the joining script via observation or trial-and-error, which can be cumbersome [17].

Analogously to our previous studies [12, 13] and Shah [31], this study distinguishes between the modification of the source code of an OSS project and the submission of this modification back to the OSS project. Newcomers have to pass both steps for a successful contribution. Both steps have their own contribution barriers, which are dubbed *modification barriers* and *submission barriers*, respectively. Accordingly, there are separate *modification motivations* and *submission motivations*.

The OSS Projects Mozilla and GNOME

Netscape made their "Communicator" an OSS project in 1998 [27]. They found the Mozilla Foundation to host the Mozilla project. While the Mozilla project is well known for its browser Firefox, it is more diverse. More than 300 modules and submodules [25] include the mail client Thunderbird, the issue tracker Bugzilla, and the cryptographic library Network Security Services, to name three examples. These applications usually target multiple platforms like Linux, MacOS, Windows, and also mobile platforms like Android. Developers may also choose among multiple development platforms including Unix and Windows [22].

GNOME [36] is a window manager and application collection for Unix operating systems. In 1997, developers in the GNU community started the GNOME project [6]. Today, the GNOME Foundation hosts the GNOME project as a non-profit organization. An elected board governs the community [10].

Mozilla as well as GNOME each use a Bugzilla instance as issue tracker [24, 35]. Both follow a process that enforces an issue in the Bugzilla instance for every change to the source code [10,23,29]. This enabled us to find first-time contributors via their issue tracker.

RELATED WORK

Von Krogh et al. [37] summarize the state of research on motivations of OSS contributors. Primarily Steinmacher et al. [33] have researched contribution barriers from different angles. They also discuss positive aspects of contribution barriers.

Recently, Lee et al. [20] analyzed motivations and entry barriers of newcomers with a survey in a similar methodology as in this study. They looked at barriers that prevent One-Time code Contributors from becoming Long-Term code Contributors, though, and not the initial contribution barriers a newcomers have to overcome for their first contribution.

Herraiz et al. [15] analyzed the joining scripts of the GNOME project [36]. They recognized that there are multiple different ways to join GNOME as a developer. In particular, 7 of 8 analyzed volunteers joined GNOME adhering the Onion Model of Joining OSS projects [44]: They first post a message on the mailing list and only afterwards they report a bug, submit a source code modification, and eventually gain commit rights to the VCS. All members of a second group of 12 employees and university staff members used a different sequence, starting almost simultaneously with a message in the mailing list, bug reports, source code submissions, and VCS commits. However, the results may be seen as debatable, since the raw data presented in the paper seem to contain exceptions to those sequences described in their text.

Still, the study shows that OSS projects may have multiple different joining scripts. It is not entirely clear yet what types of joining scripts exist and what types of OSS projects have which joining scripts.

Joining script determine the interaction of newcomers with the OSS project and therefore which contribution barriers the newcomers may possibly face. Hence knowledge about joining scripts allows deductions about which contribution barriers are relevant in a given OSS project. Even if a contribution barrier appears in a specific OSS project, it is also unclear as of yet how strong the impact of each individual contribution barrier is on each newcomer.

Weiss et al. [40] showed that developers sometimes join OSS projects in groups: there are cases where multiple developers that collaborate in one OSS project join another OSS project together.

Jergensen et al. [18] found evidence that the majority of OSS developers dedicatedly use the VCS system, and no social mediums like mailing lists. One explanation they proposed is that these developers may have socialized in related OSS projects and therefore were already well-known in the community of the OSS project. This indicates that prior exposure to OSS projects reduces the effect of social contribution barriers.

Bird et al. [2] found that joining an OSS project requires the acquisition of project-specific skills. This acquisition takes time. Table 1 shows the median times newcomers need for their first patch submission and acceptance for the three OSS projects Bird et al. had analyzed. They also showed that the motivation to join an OSS project declines over time after the initial contact. Although the study did not explicitly state this,

OSS project	Median time after the first email to the mailing list until …	
	… first patch submission	… first patch acceptance
Postgres	2. month	3. month
Apache	2. month	10. month
Python	6. month	13. month

Table 1. Time needed to acquire project-specific skills [2]

consequentially OSS projects that require long naturalization will gain less new developers.

SURVEY DESIGN

In a previous exploratory survey [12], we asked professional developers to report about contribution barriers experienced when working on OSS projects. We based the survey presented in this article on the exploratory survey, but targeted developers who had recently joined an OSS project. The survey design founded on the Tailored Design Method (TDM) [8] with some adaptations to accommodate to the survey's circumstances, especially invitations via email. A previous paper detailed insights about newcomers motivations gained from the same survey [13].

The survey targeted developers whose first patch was recently accepted by one of the OSS project Mozilla or GNOME. We selected Mozilla and GNOME because their bug-driven development allowed us to find all newcomers, as detailed in the next paragraphs.

A data collection and filtering process, depicted in Figure 1 for Mozilla, selected the participants: Scripts regularly downloaded the bug data from Mozilla's and GNOME's bug trackers. The scripts assigned all accepted patches to developers. For each developer, the chronologically first acceptance of a patch indicated the date when this person, by this survey's definition, became a developer of the project. Participant candidates are those who became developers between April 2013 and October 2013 for Mozilla and between November 2013 and January 2014 for GNOME. Afterwards, two thirds of the participant candidates had to be filtered out, because they were employees of Mozilla or the scripts had incorrectly identified them as first-time contributors. In two more cases, we could not find a valid email address of the contributor.

After filtering, 190 first-time contributors remained as invitees to answer an online questionnaire. Invitations were sent at different dates, so the invitees' memories about their first patch were still fresh. We contacted the invitees up to three times via email. The emails included one $2 gift code for amazon.com for each invitee, according to the TDM. In spirit of the TDM, the invitation emails varied in appearance: The first email had a plain text format, the second email had a more fancy Hypertext Markup Language (HTML) format, and the third email was plain text again, but signed with a digital certificate from the University of Duisburg-Essen's Certification Authority (CA). Each first email further included a manually written short summary of the invitee's first accepted patch to the project. The summary proved that the invitation is no mass mailing but a personal invitation. A link to the issue for which the patch was submitted accompanied the summary. The survey referred to this specific patch in some of the questions and the link served as a reminder and clarification which patch the survey was referring to – this ensured that the selection procedure had correctly identified the first patch and also clarified cases where it is not immediately clear to the invitee which patch was the first.

A total of 97 Mozilla contributors and 31 GNOME contributors responded to the survey. 6 Mozilla respondents and 5 GNOME respondents were filtered out, mostly because they had answered only very few questions. The 91 and 26 useful responses out of 132 and 48 invitations for Mozilla and GNOME, respectively, results in a response rate of 68.8 % and 54.2 %, in total exactly 65 %. This response rate is within the typical range of well-designed surveys [8, p. 3f]. More importantly, the response rate is higher than all other surveys targeting OSS developers to the best of our knowledge, where response rates have been at most 38.1 % [1, 14, 19–21, 41, 42].

Occupation

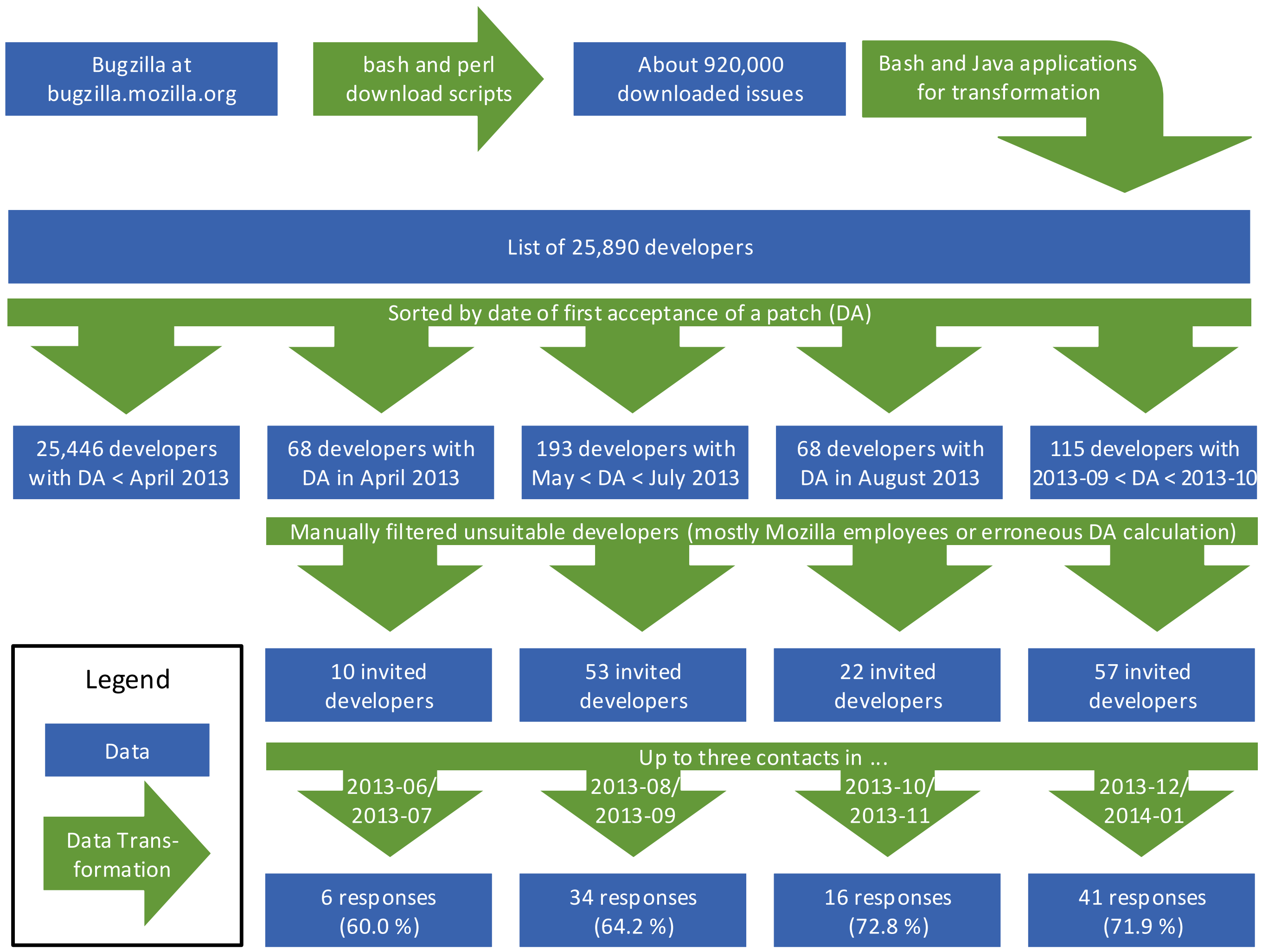
The questionnaire asked the participants about their current occupation. In terms of occupation, newcomers to Mozilla and GNOME are surprisingly similar, as Fig. 2 shows. About 49.0 % of the respondents are employees. Mozilla employees were not invited to the survey, as they do not approach the OSS project from the outside. However, the questionnaire did not ask whether employed respondents contributed to the OSS project during their free-time or whether their contribution was part of their paid work. The latter group is shown to be about half of all contributors to OSS projects [30], but not necessarily half of the newcomers in a given time frame. Another 36.5 % of the participants are students.

This partition of occupations is similar to results from other surveys of OSS developers not restricted to newcomers. However, the ratio of students is higher in this survey: Only 14 % of Hars and Ou's respondents from various OSS projects were students [14]. 23 % of the Linux kernel developers in 2000 were students [16]. 19.5 % of 684 OSS developers working on projects hosted on SourceForge.net [32] in 2001 were students [19]. Of the 1488 OSS developers whom David et al. surveyed in 2003, 28.8 % were students [5]. A later sample of 148 OSS developers, mainly from SourceForge.net, included at most 13 % students [41]. In a more recent survey, 13.1 % of 848 R package authors responded that they were students [21].

There are three important differences between the target population of prior research with its lower number of students among the contributors compared to the participants of this survey:

First, the existing research is mostly about 10 years older. The structure of the OSS contributor population may have changed in the meantime. Future researchers should try to reproduce these earlier results to decide this hypothesis.

Second, the difference might be a peculiarity of Mozilla and GNOME. Mozilla employs student programs like Google Summer of Code [26] and is the subject of some OSS university courses as answers to other questions in this survey show.



Bugzilla at bugzilla.mozilla.org
bash and perl download scripts
About 920,000 downloaded issues
Bash and Java applications for transformation
List of 25,890 developers
Sorted by date of first acceptance of a patch (DA)
25,446 developers with DA < April 2013
68 developers with DA in April 2013
193 developers with May < DA < July 2013
68 developers with DA in August 2013
115 developers with 2013-09 < DA < 2013-10
Manually filtered unsuitable developers (mostly Mozilla employees or erroneous DA calculation)
10 invited developers
53 invited developers
22 invited developers
57 invited developers
Up to three contacts in ...
2013-06/ 2013-07
2013-08/ 2013-09
2013-10/ 2013-11
2013-12/ 2014-01
6 responses (60.0 %)
34 responses (64.2 %)
16 responses (72.8 %)
41 responses (71.9 %)
Legend
Data
Data Transformation

Figure 1. Data flow for the selection and invitation of survey participants from Mozilla

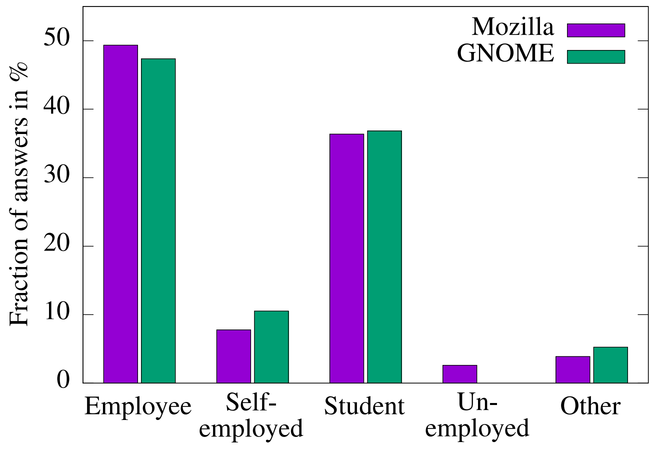


Figure 2. Occupation of newcomers

However, this cannot explain all of the difference and it does not explain why GNOME also has a comparably high number of students. The difference is therefore unlikely to be project-specific, especially because earlier research confirmed their results in a wide range of different projects.

Third, this survey targets newcomers while the other research targets OSS developers in general. There are two explanations of why this difference in group structure should create a difference in the fraction of students. Firstly, those who are students on their first contribution may have become employees after a while and may show up as employees in later surveys. This would mean that a considerable fraction of developers choose their OSS projects as students and then stick with the project. However, Hertel et al.'s participants stayed with the project only for 17 months on average [16], which contradicts that contributors stick for a longer time with an OSS project after they finish their university studies. Secondly, maybe students contribute only for a short period and then drop out of the project, while employed contributors stick with the project after their first contributions. Thus, later samples of OSS contributors would cover the employed contributors who are still with the project, but not the students who had left after their first contributions already. Which, if any, of these two explanations is correct, should be the subject of future research.

CONTRIBUTION BARRIERS

As explained initially, the study distinguishes modification and submission barriers as well as modification and submission motivations. The questionnaire asked for each of the four individually, first with an open question and then with a closed question.

In case of contribution barriers, the closed question prompted the participants to rate each predefined contribution barrier on a five-point-scale from "no obstacle at all" to "almost a show stopper". The predefined contribution barriers represent typical steps that a newcomer has to go through.

For each of the two types of motivation, the participants could pick motivations that applied for them from a list, and put them into an order that represented the importance of the selected motivations. Results for the motivation questions were analyzed in more detail in a previous study [13].

For reasons of brevity, results for the motivation questions and for the open questions on contribution barriers will not be discussed here, as only the closed questions were used to test the research hypothesis.

Modification Barriers

Table 2 lists the answers for modification barriers. Each column represents a predefined modification barrier. The table lists how many respondents rated each modification barrier at each level of importance, where 5 ("almost a show stopper") corresponds to the highest and 1 ("no obstacle at all") to the lowest importance. The bottom row lists an arithmetic mean for each modification barrier, calculated by interpreting the five levels of importance as the numbers 1 to 5.

The modification barrier *Find the Code* received the greatest arithmetic mean rank. The large size of Mozilla's and GNOME's code base impede locating the correct place to modify the source code for a specific change. The second-most important modification barrier were problems with the *Build* of the project. *Setup* represents "difficulties installing the development environment". Interestingly, writing the actual code for the bug or feature, the *Solution* of the problem, has a lower mean rank than the former three modification barriers. Contrary to the other modification barriers, nobody found the Solution to be a "show stopper" (rank 5).

Bug Reproduction comprises issues to find inputs on which the application fails. Respondents seem to underestimate the danger of *Redundant Work*, "concerns of wasted modification effort, as someone else might work on the task in parallel": Gousios et al. showed that the reason for 43 % of all patch rejections on GitHub were redundant work on the same problem [11]. The modification barrier *Download* lets participants rank the "difficulties downloading the right version of the source code". The least important modification barrier for the respondents are problems with *Community* Support.

Submission Barriers

Analogously to Table 2 for modification barriers, Table 3 lists the answers to the closed question on submission barriers. The submission barrier *Submission Procedure* has a relatively high mean of 1.980, making it the highest overall contribution barrier after Find the Code. When only considering the levels 4 and 5, Submission Procedure is ranked even higher than Find the Code and all other contribution barriers.

This indicates that although the Submission Procedure works for a considerable fraction of contributors, it is a high contribution barrier for some. Respondents explained high rankings with difficulties using the VCS in the way the OSS project expects them to. Partly, this seems to be a matter of personal preference; one Mozilla newcomer explained that "patch-based submissions is [sic] a lot of effort compared to pull requests" as they are common on GitHub. On the other hand, another newcomer explained the selected importance of 4 with "little experience with git" used for the Mozilla Webmaker component, to which this newcomer had contributed.

With an arithmetic mean rank of 1.710, issues with the *Documentation*, called "Instructions on homepage" in the question, were on par with the modification barrier Solution. The *Issue Tracker* was Bugzilla for both projects. Respondents explained in comments that Bugzilla dictates parts of the submission process that was especially difficult. This could also be seen as an issue of the Submission Procedure. Thus, there are only minor submissions problems distinctively caused by the Issue Tracker. "*Bureaucracy*/Paperwork" are usually not an issue for the participants. The participants could also rate whether the "attitude of project members" constitutes a problem for them. This *Member Attitude* received the lowest mean of 1.222 among the predefined submission barriers. Three answers in the category *Other* were reassigned to Submission Procedure, as these participants explained their choice with problems of patch file creation and the VCS, which other respondents regarded as a problem of the Submission Procedure.

ANALYSIS

According to the initial hypothesis, motivations indicate the expectations of newcomers, and contribution barriers have a lower impact on contributors who expect these contribution barriers. To test this hypothesis, we assigned roles to the participants depending on their motivation. The questionnaire included one question for each modification and submission motivation that let participants select and rank motivations from a predefined set.

Technology Learners is an example of a motivation role used for the analysis. Technology Learners are those participants who ranked learning as the most or second-most important modification motivation. Each participant can be in multiple motivation roles. Motivation roles presuppose that the participants had a specific model about the contribution and that they should have expected or not expected a particular hurdle in the contribution procedure.

Technology Learners should have expected that the OSS project's technical environment has some intricacies that a prospective contributor has to overcome – by learning about

Importance	Download	Setup	Build	Bug Repro.	Redundant Work	Find the Code	Solution	Community	Other
5	1	1	1	2	2	2	0	1	1
4	2	8	5	3	3	8	3	1	0
3	7	11	18	7	6	24	13	9	0
2	17	24	23	22	16	29	35	13	1
1	71	54	48	57	65	36	47	69	70
Mean	1.418	1.755	1.821	1.582	1.489	2.101	1.714	1.409	1.069

Table 2. Number of answers for each level of difficulty and modification barrier in the closed questions for modification barriers

Importance	Submission Procedure	Documentation	Issue Tracker	Bureaucracy	Member Attitude	Other
5	4	0	0	1	2	0
4	6	6	1	2	0	0
3	21	12	11	6	1	0
2	22	24	24	13	12	1
1	47	51	62	65	84	63
Mean	1.980	1.710	1.500	1.402	1.222	1.016

Table 3. Number of answers for each level of difficulty and submission barrier in the closed questions for submission barriers

the OSS project's technology. These hurdles manifest in contribution barriers that the survey measures. Differences in perception of these contribution barriers between members and non-members of a motivation role support the hypothesis, while a lack of differences opposes it.

Table 4 lists all motivation roles. The second column lists the number of members and non-members of the motivation role and which contribution barriers should be lower or greater for members of the motivation role according to the hypothesis. The last column lists p-values that result from one-sided t-tests that test whether the expected difference between members and non-members of the motivation role exist. For most motivation roles, multiple different contribution barriers are tested for differences resulting in multiple p-values. Using Wallis' method of combining p-values from multiple experiments [39], these p-values are combined into a single p-value for each motivation role and eventually for the whole comparison.

Each motivation role has a complementary role that comprises all respondents who are not in the motivation role. The complementary role to Technology Learners comprises the respondents who rejected learning as a modification motivation or ranked it only third-most important or less. There are 44 Technology Learners and 50 respondents in their complementary role. The null hypotheses are: Technology Learners rank the contribution barriers Setup and Build at least as high as their complementary role. As the result of t-tests, there is a statistically significant difference between Technology Learners and their complementary role for Setup and there is no statistical significance for Build.

Analogously, the motivation role *Joy Programmers* includes those respondents who ranked the joy of programming as their primary modification motivation. We hypothesize that those who want to experience the joy of programming know that programming is a challenge and specifically want to face it. The differences are only slightly significant ($p < 0.1$).

Pragmatic Patchers are those respondents who wanted to use their modification for themselves as their primary or secondary modification motivation. Pragmatic Patchers supposedly consider patching the application as less effort than a workaround like using another application, as that would also satisfy their motivation. Accordingly, they should not expect a high effort for their contribution. Core contribution barriers like Solution are obviously necessary for the modification and therefore less likely to be underestimated; thus, we operationalize the unexpected effort via the secondary contribution barriers Download, Setup, and Build. However, there are no statistically significant differences for these contribution barriers.

We wanted to shed light on the lack of differences between Pragmatic Patchers and their complementary role with a post hoc analysis. The *Strictly Pragmatic Patchers* include only those respondents who ranked Own Need as a primary motivation and not those who ranked it second. The contribution barriers for the Strictly Pragmatic Patchers were compared with those of the respondents who saw Own Need as no motivation at all. This corresponds to two of the three groups identified in our earlier research on newcomers [13]. Additionally, the post hoc analysis includes the contribution barriers Find The Code and Solution, as these are among the highest modification barriers according to the closed question for modification barriers, and thus are more likely to be the source of an unexpectedly high modification effort.

Table 5 lists the results for this post-hoc analysis. Just like the three contribution barriers already tested, Find the Code and Solution were not significantly different between Pragmatic Patchers and their complementary role (p=0.7197 and p=0.2015). The differences between the Strictly Pragmatic Patchers and those who had no Own Need at all were only

Motivation role	Role sizes (respondent/ complementary)	Contribution barrier	Expected Difference	p-value
Technology Learners	44/50	Setup	$<$	0.0490*
		Build	$<$	0.2319
		Combined		**0.0623**†
Joy Programmers	35/59	Find the Code	$<$	0.1728
		Solution	$<$	0.0518†
		Combined		**0.0511**†
Pragmatic Patchers	29/65	Download	$>$	0.2478
		Setup	$>$	0.7426
		Build	$>$	0.4598
		Combined		**0.5517**
Community Joiners	36/58	Community Support	$<$	0.0526†
		Combined		**0.0526**†
OSS Learners	42/49	Submission Procedure	$<$	0.2052
		Bug Tracker	$<$	0.9671
		Bureaucracy	$<$	0.3687
		Combined		**0.5146**
Groups combined				**0.0301***

† significant at $\alpha = 0.1$
* significant at $\alpha = 0.05$

Table 4. Which motivation roles perceive contribution barriers differently than their complementary roles, as p-value of a *one-sided* t-test

slightly significant for Download (p=0.08887) and not significant for Setup (p=0.2412), Build (p=0.2215), Find the Code (p=0.1586), and Solution (p=0.1035). However, combining these p-values using Wallis' method [39] yields a significant difference (p=0.04129) between Strictly Pragmatic Patchers and the respondents without Own Need. This result is even more outstanding when considering that the roles of Strictly Pragmatic Patchers and respondents without Own Need were smaller than Pragmatic Patchers and their complementary role: Although statistical tests generally yield lower p-values for larger groups if an effect exists, Strictly Pragmatic Patchers still show statistically detectable differences to respondents without Own Need. This indicates that the a priori differentiation between Pragmatic Patchers and their complementary role did not very well match real differentiations between motivation roles of contributors. Strictly Pragmatic Patchers and respondents without Own Need better characterize motivation roles than Pragmatic Patchers and their complementary role.

We call those respondents who ranked contact with the community as a modification motivation on first or second *Community Joiners*. Community Joiners expect social interaction with the community of the OSS project. They want to get in contact not purely because of an immediate need, instead social contacts have value for them in their own right. Their complementary role on the other hand may find it unexpected if it is necessary to get in touch with multiple members of the community in order to modify code and submit the code change.

OSS Learners are the respondents whose first or second submission motivation was learning. This is different to learning as modification motivation in the definition of Technology Learners. OSS Learners know that each OSS project has its own processes and its own culture, because their motivation is to learn about these specifics. They supposedly expect submission effort and therefore do not experience it as a submission barrier. However, neither of the three submission barriers Submission Procedure, Bug Tracker, and Bureaucracy show statistically significant differences between OSS Learners and their complementary role. Possibly, both knew about and expect the efforts of the submission, but only the former saw it as a motivation for submission, while the latter just found it to be a necessary task.

In summary, we analyzed five motivation roles, three of which had statistically slightly significant differences. The remaining two motivation roles Pragmatic Patchers and OSS Learners showed no statistical significant differences. However, a post hoc analysis revealed that adapting the motivation role of Pragmatic Patchers to Strictly Pragmatic Patchers yields a significant difference. The results for the five motivation roles, excluding the post hoc analysis of Strictly Pragmatic Patchers, combine into a p-value of 0.0301 using Wallis' method [39]. This result supports the initial hypothesis that contribution barriers are less important for contributors whose motivations suggest that they had expected them.

A NEW MODEL FOR JOINING OSS PROJECTS

The overall impact of all contribution barriers on a newcomer is determined by which contribution barriers occur and their individual impact. Both factors can be assessed individually:

Motivation role	Role sizes (respondent/ complementary)	Contribution barrier	Expected Difference	p-value
Pragmatic Patchers	29/65	Find the Code	>	0.7197
		Solution	>	0.2015
Strictly Pragmatic Patchers	23/30	Download	>	0.0888[†]
		Setup	>	0.2412
		Build	>	0.2215
		Find the Code	>	0.1586
		Solution	>	0.1035
		Combined		**0.0412**[*]

[†] significant at $\alpha = 0.1$
[*] significant at $\alpha = 0.05$

Table 5. Post hoc analysis of Strictly Pragmatic Patchers with p-values of *one-sided* t-tests

First, which contribution barriers take effect? Von Krogh et al. argued that newcomers have to adhere a specific process called the joining script to become co-developers of an OSS project [38]. Jensen and Scacchi noted that there are different joining scripts in an OSS project [17]. Herraiz et al. [15] specifically showed that GNOME employs at least two joining scripts. Moreover, volunteer developers of GNOME use one joining script while employed developers use another. Since every joining script comes with its own steps and tasks, the joining script defines which contribution barriers are in effect.

Second, what is the impact of each contribution barrier? All newcomers have their own mental model of the OSS project. The mental model determines the impact of each individual contribution barrier. If newcomers need more time to carry out a task, this amplifies contribution barriers that come with the task, but it is more important whether the newcomers expect the task. As shown in the Analysis, the mental model influences motivations and thus motivations are indicators of the mental model.

Social properties of newcomers influence both factors. For example, learning is an important motivation for students. For Technology Learners, the modification barriers for setting up a development environment have less impact. Students are volunteer developers and therefore advance slowly through supporting roles in the OSS project before they contribute code [15]. Barriers that hinder non-code contributions would therefore indirectly hinder code contributions from students.

The consequences of this insight on the management of each OSS project depends on the project's goals. It could be reasonable to lower particularly those contribution barriers that strongly affect the contributors from social groups most attracted by the OSS project. This would maximize developer influx. Inversely, an OSS project might increase attractiveness for contributors of social groups that the OSS project has low contribution barriers already.

Section Occupation discusses evidence that some social groups like non-students stay longer with an OSS project than others. Lowering contribution barriers for those who contribute to the OSS project over a long period may be desirable.

Another approach would be lowering contribution barriers for underrepresented social groups. This would diversify the OSS project's group of developers and thereby help to implement requirements of users belonging to the previously underrepresented social groups.

THREATS TO VALIDITY

This section discusses different types of threats to the survey and how the survey set-up ensured validity in spite of these threats.

Incomprehensible or ambiguous questions endanger construct validity, as the participants' answers would not fit to the questions. However, both the exploratory and the main survey received thorough pretests and the first participants were invited only after the pretesters had not misunderstood any question. The participants' answers also showed few signs of misunderstanding and those were discussed in the description of results.

The analysis frequently uses t-tests on scores from ordinal scales similar to Likert scales to prove or disprove statistical hypotheses. t-tests require normally distributed variables, which is never the case for Likert scores. Nevertheless, the t-test is robust to minor violations to its requirements [28] and works well especially with Likert scores [7].

There may be a self-selection bias, as the set of survey invitees and survey participants are not identical. However, the response rate of 65 % is high for a survey [8, p. 3f], which reduces the likelihood of a self-selection bias.

Both Mozilla and GNOME are very successful OSS projects. As the success of an OSS project depends on the project's ability to gain new developers [3, 43], both GNOME and Mozilla supposedly have a relatively low level of contribution barriers. Other OSS projects may have a generally higher level of contribution barriers.

Furthermore, each OSS project's community structure is different. The results of this survey depend on Mozilla's and GNOME's recruiting strategies, governance, and possibly specific events in Mozilla and GNOME during the study period.

This is not so much of a threat against the new model for joining OSS projects, though: We constructed the Theory from previous results and then picked two OSS projects, which provided evidence for the truth of the theory, without any evidence in advance that these two specific OSS projects would be different than others in regards to the theory.

CONCLUSION

Based on the Theory of Dissonance [9], we hypothesized that contribution barriers would impact those newcomers to OSS projects stronger that do not expect them. Whether a newcomer expects a contribution barrier would show up in the newcomer's motivation for the contribution, as both expectancy and motivation derives from the newcomer's mental model about the OSS project.

With a web-based survey of 117 newcomers to the OSS projects Mozilla and GNOME, we collected data about the effect of contribution barriers on newcomers and their contributor motivations. These data allowed a statistical analysis of the hypothesis. The results confirmed the hypothesis.

This lead to a new model of joining OSS projects linking social groups of newcomers, motivations, and contribution barriers. OSS projects may adapt their joining scripts to the social groups they want as contributors.

REFERENCES

1. Hind Benbya and Nassim Belbaly. 2010. Understanding Developers' Motives in Open Source Projects: A Multi-Theoretical Framework. *Communications of the AIS* 27 (Jan. 2010), 589–610.
2. Christian Bird, Alex Gourley, Premkumar Devanbu, Anand Swaminathan, and Greta Hsu. 2007. Open Borders? Immigration in Open Source Projects. In *Mining Software Repositories, 2007. ICSE Workshops MSR '07. Fourth International Workshop on*. 6.
3. Andrea Capiluppi and Martin Michlmayr. 2007. From the Cathedral to the Bazaar: An Empirical Study of the Lifecycle of Volunteer Community Projects. In *OSS2007: Open Source Development, Adoption and Innovation (IFIP 2.13)*, Vol. 234/2007. Springer, 31 – 44.
4. Indushobha Chengalur-Smith, Anna Sidorova, and Sherae L. Daniel. 2010. Sustainability of Free/Libre Open Source Projects: A Longitudinal Study. *Journal of the Association for Information Systems* 11, 11 (Nov. 2010), 657–683.
5. Paul A. David, Andrew H. Waterman, and Seema Arora. 2003. FLOSS-US – The Free/Libre & Open Source Software Survey for 2003. (Sept. 2003). `http://www-siepr.stanford.edu/programs/OpenSoftware_David/FLOSS-US-Report.pdf` [accessed 2017-03-27].
6. Miguel de Icaza. 1997. The GNOME Desktop project. email to the GTK mailing list. (Aug. 1997). `https://mail.gnome.org/archives/gtk-list/1997-August/msg00123.html` [accessed 2017-03-23].
7. Joost C. F. de Winter and Dimitria Dodou. 2010. Five-Point Likert Items: t test versus Mann-Whitney-Wilcoxon. *Practical Assessment, Research & Evaluation* 15, 11 (Oct. 2010), 1–16.
8. Don A. Dillman. 1999. *Mail and Internet Surveys: The Tailored Design Method*. Wiley.
9. Leon Festinger. 1957. *A Theory of Cognitive Dissonance*. Stanford University Press.
10. GNOME Wiki. 2015. Community. (Dec. 2015). `https://wiki.gnome.org/Community` [accessed 2017-03-23].
11. Georgios Gousios, Martin Pinzger, and Arie van Deursen. 2014. An Exploratory Study of the Pull-based Software Development Model. In *Proceedings of the 36th International Conference on Software Engineering*. ACM, New York, NY, USA, 345–355.
12. Christoph Hannebauer, Matthias Book, and Volker Gruhn. 2014. An Exploratory Study of Contribution Barriers Experienced by Newcomers to Open Source Software Projects. In *Proceedings of the 1st International Workshop on CrowdSourcing in Software Engineering*. ACM, New York, NY, USA, 11–14.
13. Christoph Hannebauer and Volker Gruhn. 2016. Motivation of Newcomers to FLOSS Projects. In *Proceedings of the 12th International Symposium on Open Collaboration (OpenSym 2016)*. ACM.
14. A. Hars and Shaosong Ou. 2001. Working for free? Motivations of participating in open source projects. In *System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference on*. 1–9.
15. Israel Herraiz, Gregorio Robles, Juan José Amor, Teófilo Romera, and Jesús M. González Barahona. 2006. The Processes of Joining in Global Distributed Software Projects. In *Proceedings of the 2006 International Workshop on Global Software Development for the Practitioner*. ACM, New York, NY, USA, 27–33.
16. Guido Hertel, Sven Niedner, and Stefanie Herrmann. 2003. Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel. *Research Policy* 32, 7 (2003), 1159–1177. Open Source Software Development.
17. Chris Jensen and Walt Scacchi. 2007. Role Migration and Advancement Processes in OSSD Projects: A Comparative Case Study. In *29th International Conference on Software Engineering*. IEEE Computer Society, Los Alamitos, CA, USA, 364–374.
18. Corey Jergensen, Anita Sarma, and Patrick Wagstrom. 2011. The Onion Patch: Migration in Open Source Ecosystems. In *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering (ESEC/FSE '11)*. ACM, New York, NY, USA, 70–80.
19. Karim Lakhani and Robert Wolf. 2005. *Perspectives on Free and Open Source Software*. MIT Press, Cambridge, Chapter Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects., 1–27.

20. Amanda Lee, Jeffrey C. Carver, and Amiangshu Bosu. 2017. Understanding the Impressions, Motivations, and Barriers of One Time Code Contributors to FLOSS Projects: A Survey. In *Proceedings of the 39th International Conference on Software Engineering (ICSE '17)*. IEEE Press, Piscataway, NJ, USA, 187–197.

21. Patrick Mair, Eva Hofmann, Kathrin Gruber, Reinhold Hatzinger, Achim Zeileis, and Kurt Hornik. 2015. Motivation, values, and work design as drivers of participation in the R open source project for statistical computing. *Proceedings of the National Academy of Sciences* (2015).

22. Mozilla Developer Network. 2016. Build Instructions. (June 2016). `https://developer.mozilla.org/en-US/docs/Mozilla/Developer_guide/Build_Instructions` [accessed 2017-03-23].

23. Mozilla Developer Network. 2017. How to Submit a Patch (Preparation). (Jan. 2017). `https://developer.mozilla.org/en-US/docs/Mozilla/Developer_guide/How_to_Submit_a_Patch#Preparation` [accessed 2017-03-23].

24. Mozilla Foundation. 2001. Bugzilla@Mozilla. (Aug. 2001). `https://bugzilla.mozilla.org` [accessed 2017-03-23].

25. Mozilla Wiki. 2015. Modules. (May 2015). `https://wiki.mozilla.org/Modules` [accessed 2017-03-23].

26. Mozilla Wiki. 2017. SummerOfCode. (March 2017). `https://wiki.mozilla.org/SummerOfCode` [accessed 2017-03-23].

27. Netscape Communications Corporation. 1998. Netscape Announces mozilla.org, a Dedicated Team and Web Site Supporting Development of Free Client Source Code. (Feb. 1998). `http://wp.netscape.com/newsref/pr/newsrelease577.html` [original, broken], `https://web.archive.org/web/20021004080737/wp.netscape.com/newsref/pr/newsrelease577.html` [Internet Archive, accessed 2017-03-23].

28. Geoff Norman. 2010. Likert scales, levels of measurement and the "laws" of statistics. *Advances in Health Sciences Education* 15, 5 (2010), 625–632.

29. Christian Robotoom Reis and Renata Pontin de Mattos Fortes. 2002. An Overview of the Software Engineering Process and Tools in the Mozilla Project. In *Proceedings of the Open Source Software Development Workshop*. 155–175.

30. Dirk Riehle, Philipp Riemer, Carsten Kolassa, and Michael Schmidt. 2014. Paid vs. Volunteer Work in Open Source. In *Proceedings of the 47th Annual Hawaii International Conference on System Sciences*. Computer Society Press.

31. Sonali K. Shah. 2006. Motivation, Governance, and the Viability of Hybrid Forms in Open Source Software Development. *Management Science* 52, 7 (2006), 1000–1014.

32. Slashdot Media. 2017. SourceForge Website. (March 2017). `https://sourceforge.net` [accessed 2017-03-23].

33. Igor Steinmacher, Tayana Conte, Marco Aurélio Gerosa, and David Redmiles. 2015. Social Barriers Faced by Newcomers Placing Their First Contribution in Open Source Software Projects. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing (CSCW '15)*. ACM, New York, NY, USA, 1379–1392.

34. Igor Steinmacher, Marco Aurélio Gerosa, and David F. Redmiles. 2014. Attracting, Onboarding, and Retaining Newcomer Developers in Open Source Software Projects. In *Workshop: Global Software Development in a CSCW Perspective*.

35. The GNOME Project. 2017a. GNOME Bugzilla. (March 2017). `https://bugzilla.gnome.org/` [accessed 2017-03-23].

36. The GNOME Project. 2017b. GNOME Website. (March 2017). `https://www.gnome.org/` [accessed 2017-03-23].

37. Georg von Krogh, Stefan Haefliger, Sebastian Spaeth, and Martin W. Wallin. 2012. Carrots and Rainbows: Motivation and Social Practice in Open Source Software Development. *MIS Quarterly* 36, 2 (2012).

38. Georg von Krogh, Sebastian Spaeth, and Karim R. Lakhani. 2003. Community, Joining, and Specialization in Open Source Software Innovation: A Case Study. *Research Policy* 32, 7 (July 2003), 1217–1241.

39. W Allen Wallis. 1942. Compounding probabilities from independent significance tests. *Econometrica, Journal of the Econometric Society* 10, 3/4 (July - Oct. 1942), 229–248.

40. Michael Weiss, Gabriella Moroiu, and Ping Zhao. 2006. Evolution of Open Source Communities. In *OSS2006: Open Source Systems (IFIP 2.13)*. Springer, 21 – 32.

41. Chorng-Guang Wu, James H. Gerlach, and Clifford E. Young. 2007. An empirical analysis of open source software developers' motivations and continuance intentions. *Information & Management* 44, 3 (2007), 253–262.

42. Bo Xu, Donald R. Jones, and Bingjia Shao. 2009. Volunteers' involvement in online community based software development. *Information & Management* 46, 3 (2009), 151 – 158.

43. Jin Xu, Yongqin Gao, S. Christley, and G. Madey. 2005. A Topological Analysis of the Open Souce Software Development Community. In *System Sciences, 2005. HICSS '05. Proceedings of the 38th Annual Hawaii International Conference on*. 198a–198a.

44. Yunwen Ye and Kouichi Kishida. 2003. Toward an understanding of the motivation of open source software developers. In *Proceedings of the 25th International Conference on Software Engineering*. 419–429.