

Are One-Time Contributors Different? A Comparison to Core and Periphery Developers in FLOSS Repositories

Amanda Lee

Department of Computer Science
University of Alabama
Tuscaloosa, AL 35248
Email: aslee1@crimson.ua.edu

Jeffrey C. Carver

Department of Computer Science
University of Alabama
Tuscaloosa, AL 35248
Email: carver@cs.ua.edu

Abstract—Context: Free/Libre Open Source Software (FLOSS) communities consist of different types of contributors. *Core contributors* and *peripheral contributors* work together to create a successful project, each playing a different role. *One-Time Contributors (OTCs)*, who are on the very fringe of the peripheral developers, are largely unstudied despite offering unique insights into the development process. In a prior survey, we identified OTCs and discovered their motivations and barriers. **Aims:** The objective of this study is to corroborate the survey results and provide a better understand of OTCs. We compare OTCs to other peripheral and core contributors to determine whether they are distinct. **Method:** We mined data from the same code-review repository used to identify survey respondents in our previous study. After identifying each contributor as core, periphery, or OTC, we compared them in terms of patch size, time interval from submission to decision, the nature of their conversations, and patch acceptance rates. **Results:** We identified a continuum between core developers and OTCs. OTCs create smaller patches, face longer time intervals between patch submission and rejection, have longer review conversations, and face lower patch acceptance rates. Conversely, core contributors create larger patches, face shorter time intervals for feedback, have shorter review conversations, and have patches accepted at the highest rate. The peripheral developers fall in between the OTCs and the core contributors. **Conclusion:** OTCs do, in fact, face the barriers identified in our prior survey. They represent a distinct group of contributors compared to core and peripheral developers.

I. INTRODUCTION

Free/Libre Open Source Software (FLOSS) projects have long been a source of interest. Contrary to conventional software projects run by companies who pay developers to work on them, FLOSS projects offer no direct compensation to the contributor. Nevertheless, developers still offer their time and expertise to contribute to these projects. As a result, FLOSS projects develop software that can compete with commercial software, such as Firefox, or even dominate the market, such as Apache's web server. FLOSS projects consist of two types of developers: *core members* who drive the project and *peripheral developers* who contribute code and fix bugs.

One-Time Contributors (OTCs), those who have successfully contributed a single code commit, are the most peripheral of the peripheral developers [5]. Some OTCs never intend

to become long-term contributors; others are influenced by factors in or outside of the project to make only one successful commit. Through fixing bugs or offering new features, these peripheral, newcomer contributors offer valuable code to the FLOSS projects. While the motivations of core and peripheral contributors have been examined before [4], [8], [9], researchers have largely neglected OTCs. However, they can provide valuable insight into those developers on the very fringes of FLOSS projects.

In our previous work, we surveyed a group of OTCs identified from a series of FLOSS projects to determine their motivations for contributing to FLOSS projects and any barriers that stood in their way. The survey results showed that OTCs' motivations were (1) extrinsic - fixing a bug that was blocking their work and (2) intrinsic - the desire to scratch an itch or to meet a requirement of their employer. While some respondents faced no barriers (they did not contribute again because there were no other bugs in their way or because their employer no longer needed the project), others reported a number of barriers that prevented them from continuing to contribute code. The foremost barrier was time, followed by the process of contributing, which was often poorly documented or very complex. Other OTCs indicated that they were ignored or that it took a long time to get their code reviewed.

In addition to our survey results, a review of the literature identified other issues that might face OTCs. These issues include a low acceptance rate and difficulties with social aspects of the project [1], [8], [11].

Therefore, the goal of this study is to gain further insight into the behavior of OTCs, validate the survey results, and understand whether research findings about peripheral developers also apply to OTCs.

We achieve that goal via data mining of a series of FLOSS repositories. While the survey only sampled the population of OTCs from the subject projects, the data mining approach provides insights based on all OTCs identified in the project repositories. Through data mining we can analyze quantitative data at a level not possible from the original survey data. The FLOSS repositories holds rich, valuable data, including code

review data, that assists in capturing the whole picture of the OTCs contribution. These results should help to validate (or refute) the survey results as well as to provide insights into the overall behavior of OTCs.

The primary contributions of this work are:

- Understanding of the differences of the level of involvement among core, peripheral, and OTC FLOSS developers;
- A comparison of the patch size and interactions during the review process among core, peripheral, and OTC FLOSS developers; and
- A description of how OTCs are treated relative to core and peripheral developers with regard to project responsiveness and acceptance rate.

The paper is organized as follows. Section II overviews background research in this area. Section III reviews the results of our motivating survey. Section IV enumerates the research questions. Section V describes the methodology. Section VI provides the results. Section VII discusses the results. Section VIII addresses threats to validity. Section IX concludes the paper.

II. BACKGROUND

At the center of FLOSS projects, the *core contributors* are those who have been with the project for a long time and contributed a large number of patches to the project [3], [15], [16]. These contributors may be contractually obligated to contribute to the project (e.g. Google pays some employees to make contributions to the Android project) or may simply enjoy contributing to the project on their own. Many contributors are hobbyists, who view their contributions to FLOSS projects as an enjoyable pastime. Core contributors are usually aware of more of the project than other contributors and may participate in setting the direction for the project [4], [10].

Next in their level of commitment to the FLOSS project are the *peripheral contributors*, who do not remain part of the FLOSS project for as long as the core contributors and contribute fewer patches during their tenure at the project. Typically, peripheral contributors: contribute bug fixes, add to the projects documentation, help refactor code, and advertise the project to obtain new contributors [8], [9]. OTCs are a subset of peripheral developers who only have one code contribution accepted to the project repository. Finally, Casual Contributors are also very peripheral contributors who have made one or fewer code contributions, but may have contributed in ways that do not directly include code [8].

Each type of FLOSS contributor sees different motivations driving their participation in FLOSS projects. For core contributors, intrinsic motivations, such as gaining enjoyment from creating code or giving back to the project, and the motivation to attract future contributors are the driving forces [10]. Conversely, for peripheral contributors, extrinsic (or need-based) motivations, such as fixing a bug that is blocking their work or adding a new feature to help their work, are the driving forces [10]. OTCs and Casual Contributors derive motivation from both intrinsic motivations, such as the enjoyment gained

from writing and contributing code, and extrinsic motivations, fixing a bug that was blocking their work or 'scratching and itch' [5], [8].

Even with these motivations, newcomers to FLOSS projects still face a number of barriers when making their first contribution. OTCs have successfully contributed code to a FLOSS project, therefore they have succeeded despite these barriers, although some may have made prior contributions that failed to be accepted. Steinmacher, et al. identified 58 barriers standing in the way of FLOSS newcomers [11]. Our survey showed that some of these barriers, such as difficulty in setting up the projects environment and being ignored, stand in the way of OTCs as well [5]. Casual contributors also shared many of the same barriers OTCs faced even when contributing only documentation or other non-code contributions [8]. These barriers can be strong enough to make a newcomer give up on committing, or to cause an OTC to stop contributing after one commit [12].

III. MOTIVATING SURVEY

This paper is primarily motivated by the results from our own previous survey on OTCs [5]. In that study, we defined OTCs as subset of peripheral developers. The goal of the survey was to *understand the impressions, the motivations, and the barriers of OTCs in FLOSS projects*, especially as it related to why OTCs had not contributed a second code patch to their respective FLOSS projects.

To gather this data, we mined 23 popular FLOSS projects that used the Gerrit code review tool. By selecting projects that used Gerrit, we ensured that all submitted patches would be reviewed prior to acceptance and ensured that the repositories had appropriate data. We used participation in the project, as captured in the code review repository, to select only participants that met the definition of an OTC (that is, had only one successful code contribution). Of the 3,500 potential respondents, 174 completed the survey.

Based on the goals of the survey and previous literature in the area, we defined the survey questions. The survey consisted of a series of demographic questions and a series of open-ended questions about their impressions of the project, motivations for contributing, and barriers they faced in contributing or that stood in the way of future contributions. The study was analyzed by hand; results from the analysis were compared between authors until the inter-rater reliability score was 1, meaning they fully agreed on the results.

The results of the study showed that most OTCs had positive impressions of the other project contributors, including that they were skilled, friendly, or helpful. However, some had negative impressions. These negative impressions included that the other project members were busy, unresponsive, or unfriendly, and that OTCs were ignored by other project members.

Regarding their motivations for participating in FLOSS, the results showed that OTCs were motivated by both intrinsic, or internal and enjoyment-based motivations, and extrinsic, or need-based motivations. While they most respondents often

contributed to fix a bug, an extrinsic motivation, the second most common motivation was the intrinsic desire to share with the community. Some respondents also contributed because they were being paid to contribute or to add a new feature that they personally needed.

Regarding barriers faced by OTCs, slightly more than half of the respondents indicated they faced barriers that caused them to give up. These barriers included: time; the process of contributing or entering the project, such as setting up the development environment or navigating git to submit a first patch; the project's license; and a lack of knowledge, whether it be programming knowledge or knowledge about the project. Of those that faced no barriers, the most common reasons for no longer contributing were that they had nothing else to contribute, their employer no longer wanted them to contribute, they never had an intention to become a long-term contributor, or they no longer used the product.

IV. RESEARCH QUESTIONS

Drawing from the background on FLOSS and OTCs, we pose a series of research questions to provide a deeper understanding of OTCs and their interactions with FLOSS projects.

First, because the primary contributions of OTCs focus on bug fixing, their patches may be smaller than those contributed by the core or peripheral contributors, as bug fixes are often small fixes [9]. The size of the patch can be seen as a measure of effort. Larger patches with more lines of code suggest that the patchwriter has exerted more effort, an indication of increased likelihood of remaining with the project [16]. If OTCs have large patches, then they may be valuable candidates for joining the project who were instead turned away by a barrier. To better understand whether there is an observable difference in the size of the patches submitted by the different types of contributors, we pose the first research question:

RQ1: *Are OTC patches smaller than other contributors' patches?*

Second, one of the more common barriers we identified in our survey (Section III) was that OTCs felt like they were being ignored or neglected [5]. Peripheral contributors can be neglected and have their patches overlooked by their peers. Additionally, newcomer contributors can face difficulties in getting their patch reviewed and accepted, due to a lack of knowledge about the best project members with whom they should interact [9], [11], [8]. These contributors often have difficulties in receiving a timely reply to requests for reviewing submitted patches, which may lead to their patches being accepted more slowly [1]. This perception of slow response and acceptance could be at odds with other research indicating that small patches are accepted more quickly [14]. If the answer to RQ1 is 'yes', that is OTCs submit smaller patches, then those patches submitted by OTCs should be accepted rather quickly.

In addition to the speed with which the patches are accepted, another aspect of being ignored or neglected is whether or not other project members participate in discussions about the

submitted patch. The more discussion about a patch (and the more people involved in those discussions) the less likely it is that that submitter will feel ignored or neglected.

To understand both of these aspects of being ignored or neglected, we pose the following research questions:

RQ2: *Are OTC patches accepted slower than other contributors' patches?*

RQ3: *Do OTCs receive fewer comments and have fewer contributors comment on their reviews than other contributors?*

Third, OTCs may make 'drive-by commits' or 'feature gifts,' in other words, commit their code blindly and not return to the project [7], [11], [13]. These developers have no idea whether the code gets accepted, and often do not care. One indication that an OTC stays engaged in the project until patch acceptance is whether they participate in the discussion resulting from the review of their submitted patch. To understand how OTCs compare with core and periphery developers in this area, we pose the following research question:

RQ4: *Do OTCs reply to their review conversations less often than other contributors?*

Finally, in our motivating survey (Section III) we found that some OTCs reported difficulties in getting a patch accepted. To determine whether OTCs had their submissions accepted at a lower rate than peripheral or core contributors, we pose the following research question:

RQ5: *Do OTCs get accepted at a lower rate than other contributors?*

V. METHODOLOGY

We obtained the data for this study by mining the repositories of the same projects used to derive the list of potential survey respondents for our motivating survey (Section III). By using the same projects, we ensured that we were drawing from the same population for the two studies. We mined the repositories from 21 FLOSS projects. These projects ranged from very large, corporate-controlled projects such as *Android* to smaller, unsponsored projects like *mahara*. We ensured that all projects were of sufficient size and maturity to have enough OTCs to make the analysis valid. Small and or immature projects would be unlikely to have enough OTCs to provide any insights [9]. In Table I we provide an overview of the relevant characteristics of the projects included in the mining study.

The repositories contain: (1) code-review data, including reviews and review comments, (2) inline comments, (3) patch data, (4) request details, and (5) data on the contributors themselves. These different types of data ensure that we can analyze data related to each of our research questions. The data in the repositories includes information about all commits that used Gerrit code-review from project initiation through May-June 2015, when we mined the data.

Based upon our definition from OTCs as those who have had only one accepted code commit, we identified *OTC contributors* as those who had one code patch marked with

TABLE I
PROJECT DEMOGRAPHICS

Project Name	First Commit	Total Commits	Developers	OTCs
android	10/21/2008	96217	3339	531
aokp	4/5/2012	24899	362	51
coreboot	6/3/2011	17088	188	34
couchbase	5/5/2010	73415	279	19
cyanogenmod	10/28/2010	128233	2752	364
gerrit_review	10/21/2008	29313	539	166
gerrithub	10/3/2013	25699	822	141
go	11/14/2014	7158	303	101
gromacs	7/26/2011	6635	81	16
gwt	10/10/2012	4558	188	72
kitware	8/21/2010	34438	393	76
libreoffice	3/6/2012	26159	482	109
mahara	3/23/2011	8480	97	12
openstack	7/25/2011	325307	5694	1027
ovirt	10/26/2011	72300	399	38
qt	6/29/2011	171131	1523	300
scilab	4/7/2010	30976	62	3
typo3	2/23/2011	72080	1026	144
whamcloud	10/6/2010	15876	212	20
wikimedia	9/7/2011	403901	1116	177
wireshark	1/17/2014	17284	289	104

a status of ‘MERGED’, indicating that it had been accepted. We identified *core contributors* as those who had contributed 2% or more of the total code in the repository. We identified *peripheral contributors* as those who contributed less than 2% of the total code, but were not OTCs. We chose 2% as the cutoff because the percentage of total code contributed usually dropped off sharply below that figure. Additionally, when we compared our results from the 2% cutoff to those from a prior social network analysis of a subset of the included projects [2], we found largely the same results. This analysis provides confidence that the chosen threshold of 2% for core contributors is valid.

To extract data from the repositories, we developed a query for each research question. To validate the queries, we first executed them on the smallest project and manually validated the results. We chose the smallest project so that hand validation would be feasible for each query. After revising the queries as needed to ensure they provided valid results, we then executed queries on the repository for each included project to extract data about core, periphery, and OTC contributors.

After mining the data, we stored the information in a spreadsheet for each project. Some projects did not follow a standard format. Therefore, where necessary, we cleaned the data before inserting into the spreadsheet to ensure a consistent format across all projects. During the cleaning process, we did not lose or change any data. We just reformatted to match our consistent format. For analysis, we used some built in Excel functions (e.g. AVERAGE and SUM) along with more complex analyses in SPSS.

VI. RESULTS

We organize the results discussion around the four primary themes in the data, which cover the five research questions. For all statistical analyses comparing the distribution across the three groups (Core, Periphery, and OTC), we use a repeated-measures one-way ANOVA. We chose the repeated-measures ANOVA because the values for Core, Periphery, and OTC all come from the same projects. After each repeated-measures ANOVA, we use the repeated-measures pair-wise t-test to perform the post-hoc comparisons between each pair of contributor groups. We used the bonferroni correction on the pair-wise t-test to reduce the chances of detecting spurious results. None of the response variables are normally distributed. Therefore, prior to performing the statistical analyses, we performed a log transformation on the data. For transparency, we have made all our raw data and analysis scripts available online.¹

A. Patch size

RQ1 focused on understanding the size of the changes made by each type of contributor. We calculated patch size by measuring the number of lines changed (inserted and deleted) in the patch. We computed the patch size for both merged (accepted) and abandoned (rejected) patches, in order to capture the total effort a contributor makes to the project, regardless of acceptance. As we show in Figure 1, there is a general decreasing trend in patch size as we move from *core contributors* to *peripheral contributors* to *OTCs*.

This result is not surprising given the nature of FLOSS projects. Core contributors usually have enough privilege to introduce new modules or features directly, which typically require larger patches. The core contributors may additionally be the founders of the project, and therefore have made relatively large commits when setting up the project. Peripheral contributors and OTCs make most of their contributions to fix bugs or edit smaller features, which typically requires smaller patches. As OTCs have only one patch accepted, it is even more likely that they are not generally adding new modules or features, so they should have a smaller average patch size.

The difference among the groups is significant ($F_{19} = 7.0675, p = .005$). The pair-wise difference between OTC and Core is also significant ($p = .003$).

In a few projects, OTCs had larger patch sizes than the peripheral or core contributors. This result may be partially due to the fact that we are using the average patch size as a metric. Since core members make a larger number of total commits, the average will decrease if a large number of small commits offset a smaller number of very large commits. Conversely, since OTCs make only a small number of total commits, if those commits are medium size, their overall average may be higher. For example, in *couchbase*, in which the OTC average patch size is larger than the core contributor average, the core contributors contributed a total of 4,837,146 insertions and 3,555,628 deletions, or 49% of all code in the project, while the OTCs only contributed 50,777 insertions and 18,942

¹http://carver.cs.ua.edu/Data/2017/ESEM_OTC/

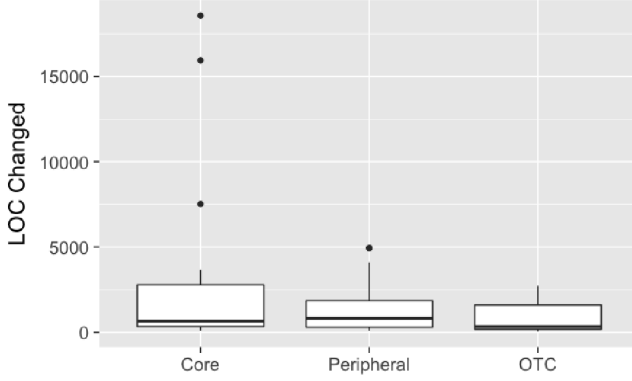


Fig. 1. Average LOC changed per commit

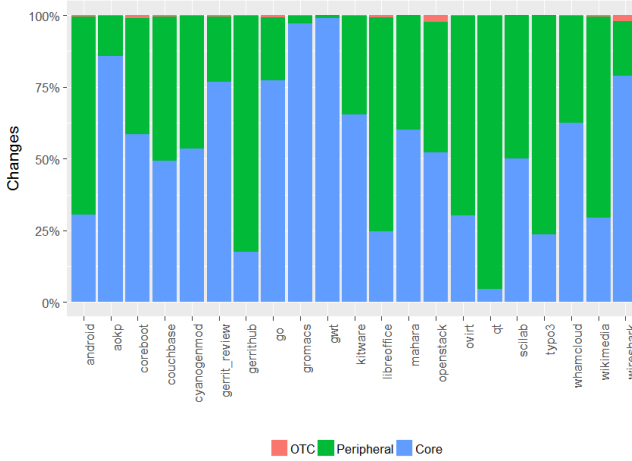


Fig. 2. Percentage of total code contributed

deletions, 0.4% of all code. The OTCs only made a total of 36 submissions, but these submissions were larger, on average, than the core submissions.

In all cases where the OTC patch size was large, there were only a small number of OTC commits. Each submission was larger, but, as we show in Figure 2, the total amount of code submitted by the OTCs was still far below the total submitted by the core contributors. On average, core developers contributed 41.5% of all code, peripheral developers contributed 58.0% of all code, and OTCs contributed 0.5% of all code.

B. Decision Interval

RQ2 focuses on the speed with which a project accepts a submitted patch. The time between when a patch is submitted and when the decision to accept or reject it is made can make or break a patch. As many patches need revisions, a long delay between submission and decision may result in the original contributor either losing interest in contributing to the project or simply forgetting about the project and moving on to another. Prior research has shown that smaller patches tend to

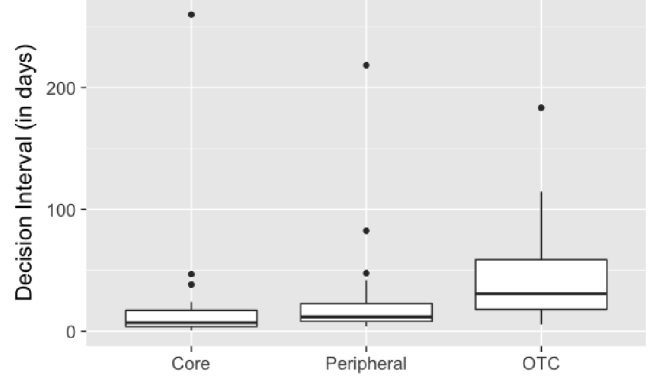


Fig. 3. Comparison of decision interval across developers in seconds

be accepted more quickly [14]. However, other research also shows that peripheral contributors can be neglected and have their contributions accepted more slowly [11], [9]. While in the discussion of RQ1 we showed that the patches submitted by OTCs are smaller on average, the goal here is to see whether they are accepted more quickly than others (as they would be if patch size was the only important factor) or if prior research holds and these OTCs are neglected as peripheral contributors.

In Figure 3 we show the average decision interval for each project across each type of contributor. Overall, there appears to be a trend, with *core contributors* having the shortest interval, followed by *peripheral contributors* and finally *OTCs*. While this general pattern is evident, it is not universal across projects. For example, in the *typo3* project (the project represented by the three outliers at the top of the figure intervals overall), the pattern is the exact opposite, with core contributors having the longest intervals and OTCs the shortest.

Looking at the whole sample, *core contributors* receive the fastest replies (approximately 23.5 days after submission). This result is expected because the core developers know the project well and can likely submit their patch to the right person so it is reviewed quickly. The average time for a peripheral developer to receive a response is slightly longer (approximately 29.2 days). Finally, at almost twice the delay faced by core contributors, OTCs face the longest interval for a decision on their submissions at approximately 46.7 days. The differences among the groups is significant ($F_{19} = 15.4239, p < .001$). All three pair-wise differences are significant: OTCs-core contributors ($p < .001$), OTCs-peripheral contributors ($p < .001$), and core contributors-peripheral contributors ($p = .003$).

Only in three projects do OTCs have a shorter average decision interval than peripheral and core developers. While these results are contrary to the earlier findings that smaller patches get reviewed faster, it is not entirely unexpected. OTCs are newcomers and may not know who to ask for a review [5], [8], [11]. Additionally, some OTCs make “drive-by commits”

and completely abandon the patch after committing it. If the drive-by patch works and requires no changes from the original author, it could get accepted. But if a reviewer requires the contributor to revise the patch before accepting it, the contributor will never reply and make the required revisions. Instead, the committer will leave the reviewer waiting until either the patch times out or the reviewer decides to abandon it.

C. Comments

RQ3 focuses on the interaction between project members and the patch submitter. It attempts to understand whether there is a difference in the number of comments that patches submitted by different types of contributors receive, as well as whether there is any difference in the number of project members that participate in the discussions about contributions from different types of contributors. As these two factors are linked (more contributors commenting requires more comments), they are discussed together.

To gather data for these questions, we processed the repositories as follows. First, we associated each comment with a patch ID. Then, for each patch ID, we counted the comments tagged with that ID. To compute the number of commenters, we counted the number of unique user IDs associated with each patch ID.

As we illustrate in Figure 4, there is a general trend that the number of comments required to resolve a patch submission increases as we move from core to periphery to OTC. The difference among the groups is significant ($F_{19} = 10.2355, p < .001$). All three pair-wise differences are significant: OTCs-core contributors ($p < .001$), OTCs-peripheral contributors ($p = .04$), and core contributors-peripheral contributors ($p < .001$).

Similarly, as we illustrate in Figure 5, there is a general trend that the number of project members who participate in the discussion increases as we move from core to periphery to OTC. The difference among the groups is significant ($F_{19} = 21.4452, p < .001$). All three pair-wise differences are significant ($p < .005$).

The patches submitted by OTCs had longer conversations that involved more people. While this result is contrary to our expectations, upon further reflection, it does make sense. The comments here are comments from the code review process about new patches. It is quite likely that other project members already know the core contributors and trust them to write good code, resulting in fewer review comments. Similarly, the project members may already know the peripheral contributors, though to a lesser extent than they know the core contributors. Hence, peripheral contributors face a more thorough review process than core contributors, though a less thorough review process than OTCs. Project members likely know very little, if anything, about the OTCs, resulting in more thorough review discussions before accepting their code submissions. Contrary to the result from RQ2 in which OTCs appear ignored, this result suggests that OTCs actually receive more attention from project members.

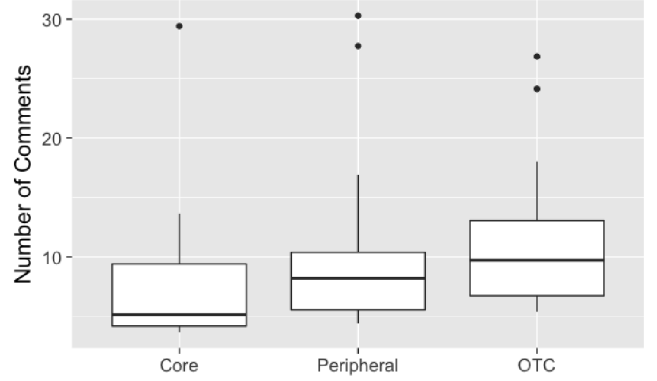


Fig. 4. Comparison of number of comments in the review conversations

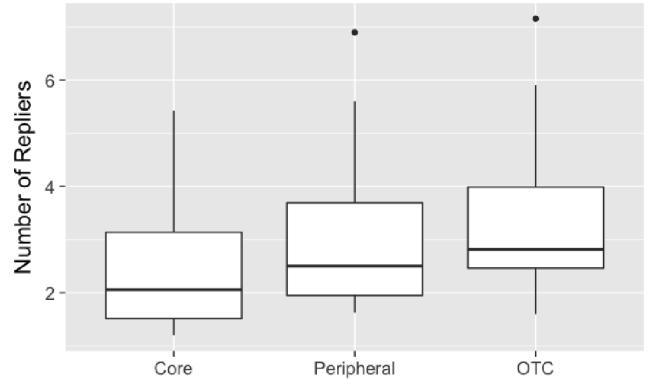


Fig. 5. Number of repliers per conversation

RQ4 focused on how involved the original contributor is during the review discussion. The original patch author was only counted as a commenter if they replied on the chain. Posting a patch generated a ‘comment,’ but as this was not a true contribution to the conversation, these were not counted as comments. Patch authors were considered as replying to their patch if their ID was captured in the conversation after the first comment.

As we show in Figure 6, the trend for this question is the inverse of the trends regarding the number of comments and number of repliers. Core contributors participated in the discussion about their submitted patches 75% of the time, peripheral members participated 70% of the time, but OTCs only participated 61% of the time. The difference among the groups is statistically significant ($F_{19} = 8.46681, p = .002$). The pair-wise differences between OTCs and core contributors ($p < .001$) and between OTCs and peripheral developers ($p < .001$) are significant.

This result may be impacted by the presence of the “drive-by commits” discussed earlier, where OTCs simply submit the code commits and do not care what happens afterwards. Another factor that might impact this result is that because

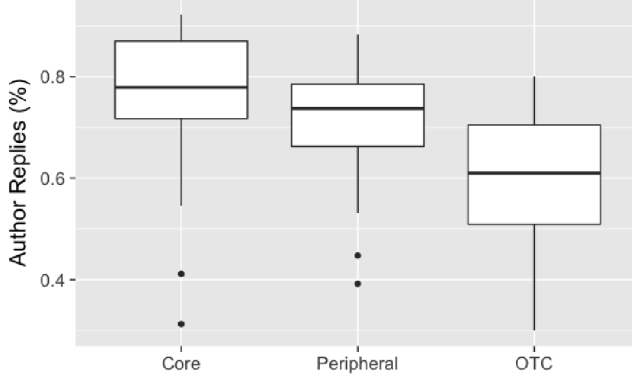


Fig. 6. Percentage of patch authors who replied

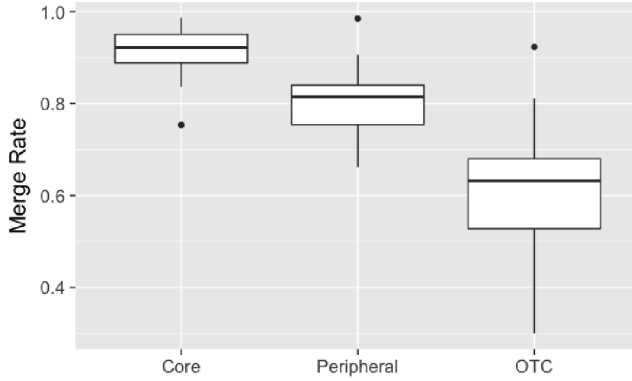


Fig. 7. Rate at which each group of developers had their code merged versus abandoned.

OTCs are new to the project, they may feel less comfortable with the other project participants and be too shy to participate in the discussion [11].

D. Abandoned and Merged Rates

To answer RQ5, we examined the merge rates of the projects. Because core contributors are well-known by the project, their patches are usually accepted at a higher rate than those of peripheral contributors [2]. We would expect patches contributed by OTCs to have the lowest acceptance rate because (a) they are the most peripheral of peripheral contributors and (b) they are newcomers to the project [11], [8]. The results for this analysis (Figure 7) confirm this expectation.

The average acceptance rate for each contributor type is: core - 91%, periphery - 80%, OTC - 62%. The overall average acceptance rate for all submissions was 86%, meaning that submissions from both peripheral and OTC contributors were accepted less frequently than the overall average. While some projects have high OTC merge rates, only one project breaks the pattern. In *gwt*, the patches submitted by OTCs are

accepted at a higher rate than those submitted by peripheral developers. Even in this case, the patches submitted by OTCs are still accepted less frequently than those submitted by core contributors.

The difference between the groups is significant ($F_{19} = 52.0501, p < .001$). Similarly, all three pair-wise comparisons are significant ($p < .001$). Based on these results, we can conclude that core contributors, peripheral contributors, and OTCs are distinct groups within the studied FLOSS projects, at least as it relates to the acceptance rate of their submitted patches. As such, acceptance rate is a strong way to differentiate core, periphery, and OTC members of a project.

Based on our definition of OTCs, while they can submit more than one patch, only one of those patches has been accepted (any others submitted have been rejected). Therefore, if the accepted patch is the only one submitted by the OTCs, then we would expect to see an acceptance rate close to 100%. However, the acceptance rate of OTC submissions is lower than for either core or peripheral contributors.

In order to understand this result better, we analyzed the rejected OTCs submissions in more detail. Most OTCs, 84%, submit exactly one patch and have had no patches rejected. Of those OTCs who have submitted more than one patch, they submit an average of 1.79 additional patches (for a total of 2.79 submitted patches). This rate of 1 commit accepted out of 2.79 patches is only a 36% acceptance rate for those that do submit more than one patch.

The most patches an OTC in our study submitted was 81 patches total, but this contributor was a clear outlier even compared to other OTCs who submitted many patches.

VII. DISCUSSION

The results of our work provide evidence that core contributors, peripheral contributors, and OTCs are distinct groups within FLOSS projects. As expected, the core contributors contribute the most code and the OTCs contribute the least, with the peripheral contributors in the middle. OTCs often remain on the fringes of projects, where decisions on the acceptance of their code contributions require more discussion over longer time intervals, resulting in lower acceptance rates compared with peripheral contributors and core contributors. However, some OTCs simply gift their code to the project and do not communicate with project members any further, which may contribute to the long decision intervals and review processes OTCs face. For all research questions, there is a significant difference between the three groups. In addition, for all research questions aside from RQ1 and the number of comments on reviews, the pairwise differences between OTCs and core and OTCs and peripheral contributors were significant. This suggests that OTCs are a distinct group of contributors that stand apart from core and from other peripheral developers.

In this section, we revisit our six research questions. Using the results described in Section VI along with previous results from the literature, we discuss the implications of the results for OTCs on FLOSS projects.

RQ1: *Are OTC patches smaller than other contributors' patches?*

Patch size decreases as we move from core contributors to peripheral contributors to OTCs. The fact that core contributors contribute the largest patches is not surprising, as core contributors often found the project [16] and often contribute entire features or initiate development of a new feature. Similarly, peripheral developers usually fix code, though they can also contribute new features [9]. Project newcomers usually make only small changes in the code and rarely contribute entire features [13]. Therefore, it is not surprising that OTCs, as newcomers, contribute the smallest patches.

Additionally, the OTCs who responded to the survey indicated their primary motivation was fixing bugs, which are often small fixes. While peripheral developers are also motivated by fixing bugs, they may choose harder bugs due to their relative higher project knowledge compared with OTCs [9], [3]. Beyond their preference to fix bugs, OTCs' short project tenure may make them hesitant about committing code or unclear on what is appropriate in the project [6], [11], [13]. The respondents to the motivating survey did not express hesitance about committing code. In fact, the OTCs who responded to the survey indicated that they had usually contributed to a different FLOSS project previously and were familiar with the concepts of contributing code and communicating with fellow contributors. However, the survey respondent OTCs did express that they had difficulties with lacking the proper knowledge to contribute, be it knowledge about the project, knowledge about the project's language, or even knowledge about coding in general [5]. Therefore, the patches submitted by OTCs may be smaller due to a difference in expertise between peripheral and OTC contributors, at least in terms of expertise on the current project. Any one of these factors working alone or in concert with other factors may lead OTCs to contribute smaller patches than other project members.

RQ2: *Are OTC patches accepted slower than other contributors' patches?*

OTCs face a significantly longer decision interval for patch acceptance than peripheral or core contributors. OTCs on average have an interval almost twice as long as core developers. This long delay is consistent with the survey response that OTCs felt ignored by project members. The delay is particularly important for FLOSS projects to address, as OTCs are aware of and affected by this discrepancy [5]. One way FLOSS projects could reduce this effect is to make their submission process more transparent and make it easier for newcomers to find a proper code reviewer [11].

This trend of a longer decision interval for OTCs is true in all but three of the studied projects. *Typo3* has the most noticeably shorter interval for OTCs (the three outliers at the top of Figure 3), but the *mahara* and *coreboot* projects also exhibit shorter intervals for OTC contributions. *coreboot* has the same pattern as *typo3* (OTC, periphery, core), while in *mahara* the pattern is OTC, core, periphery. We could not immediately identify any unique characteristics about these

three projects that would suggest different behavior. They are of different sizes, have different numbers of OTCs, and different numbers of overall contributors. We need to study these projects in more detail to understand whether there are systematic factors which could be causing the different results.

RQ3: *Do OTCs receive fewer comments and have fewer contributors comment on their reviews than other contributors?*

The data mining results showed that OTCs did not receive fewer comments or have fewer commenters than core or peripheral contributors. In fact, the OTCs had a significantly larger number of comments and contributors on their review threads. To understand review comments and answer RQ we took into account three measures to capture the quality and depth of the conversation:

- *number of comments* – indicates the depth of the conversation, with more comments indicating a deeper conversation and fewer comments suggesting that project members ignored the contributor;
- *number of repliers* – indicates the breadth of the conversation, with more replies indicating a greater interest in the conversation by more project members; and
- *whether the submitting author participated in the conversation* – indicates the investment on the part of the contributor, with an author who replies indicating interest in promoting the patch and getting it merged and an author who does not reply indicating they have lower interest and may have abandoned the patch (discussed in detail in answer to RQ5).

OTCs received more repliers and more comments than core or peripheral contributors indicating that their conversations were more in-depth. However, this result could have a negative implication, as well. OTCs likely had longer conversations with more people involved because the core and peripheral contributors have more clout in the project and have less need to explain or defend their commits. OTCs, who are not well known to project members, appear to have to defend their changes and explain their rationale more thoroughly. As the trend continues from OTC to peripheral to core, with core having the least comments and peripheral being between the two, and all differences continue to be significant, it is very likely that more review comments and commenters are considered a negative factor, as core have the least when compared to both peripheral and OTC contributors. Having said that, it does suggest that rather than being ignored, OTCs were actually given at least as much, if not more attention than other project members.

This result does not suggest that project members neglected or ignored OTCs (as the survey responses indicated). OTCs may still feel ignored despite the more in-depth participation in the review process if they do not receive the answers expected. A common negative impression expressed by survey respondents was that fellow project members were unhelpful. While this study cannot confirm such a claim, it remains that the OTCs may have been treated in such a way to feel ignored despite having more 'attention.' Also, even though there is

more participation in the discussion, the OTCs receive the overall results slower than the other contributors, which may also contribute to the perception of being ignored as they wait for comments on a submitted patch.

RQ4: *Do OTCs reply to their review conversations less often than other contributors?*

OTCs participated in the discussion about their submitted patches significantly less often than the other contributors. This result may be caused by a lack of comfort interacting with project members or lack of knowledge about how to interact. Prior research has shown that newcomers have to be socialized into FLOSS projects and face barriers related to this socialization process [6], [3], [11].

However, the lower participation rate may also result from the OTC's lack of interest and may indicate that the OTC never intended to become a long-term contributor and was only a 'drive-by' contributor [7]. The previous survey showed that over 60% of OTCs had nothing else to contribute or no intent to join the project. Meanwhile, only around 10% indicated that they faced social barriers, primarily being ignored or facing bad attitudes from the other contributors, not a fear of contributing or commenting to others. Because of these observations, it seems more likely that the OTCs are not participating in the discussion because they are not interested in joining the project rather than because they face particular social barriers.

While this reasoning may be true for OTCs, there remains a significant difference between peripheral and core developers. Peripheral developers may face the social barriers, or may be still traveling through the socialization process [6], [11].

RQ5: *Do OTCs get accepted at a lower rate than other contributors?*

OTCs and peripheral contributors have significantly lower acceptance rates than core developers, with OTCs further having significantly lower acceptance rates than peripheral contributors. This result corroborates the idea that peripheral contributors and OTCs have a more difficult time getting their patches accepted. The fact that the core contributors have the highest acceptance rate is not surprising. Core contributors usually have commit access to the code, are the best-connected project members, and know which other project members to consult if they do not have commit privileges themselves [16]. Peripheral developers usually do not have commit access, and might not be as certain who to contact for help. However, as they have committed at least two patches, peripheral contributors are at least somewhat involved in the project and will have more knowledge, in general, than an OTC, who is a newcomer.

The fact that OTCs and peripheral contributors have such different acceptance rates is somewhat surprising, since OTCs are a subset of peripheral contributors. However, peripheral contributors can actually be fairly involved in the project, with some on their way to becoming core members. While OTCs may be similar to those peripheral developers who have successfully contributed only a few patches, they will

likely be dissimilar from those peripheral developers who have successfully contributed a larger number patches. OTCs are at the very fringe of the periphery and, with only one successful commit, are usually far from becoming core members.

As OTCs are newcomers, they may not understand the proper code format or proper commit branch and may thus have their submissions rejected despite the quality of their code [3], [11]. The results of our survey supports this belief, as the second and third most common difficulties OTCs reported were with the process of contributing and difficulties entering the project. There are likely other factors which influence the low rate of OTC acceptance; however, this is the primary factor our previous survey captured.

VIII. THREATS TO VALIDITY

This section describes the various threats to validity faced in this study.

A. Construct threats

It is possible that the data we used to answer each research question does not directly correspond to the construct in that question. For example, the small size of the patches submitted by OTCs may result from barriers in the FLOSS project rather than because they preferred to fix small bugs. This threat is mitigated somewhat by the previous survey study's results. Because we analyzed data from the same FLOSS projects that we used to generate the pool of survey respondents in our prior work, we can be reasonably confident that the qualitative explanations from the survey, such as that OTCs prefer to fix bugs, can be compared with the quantitative results from the data mining in this paper. Because the mined data is drawn from the same FLOSS projects used to identify survey respondents in our previous study, we do not think there is a serious threat of the data not corresponding to the same phenomena in the survey results.

B. Internal threats

In designing the repository queries for each research question, it is possible that we did not perfectly capture the data as intended. To mitigate this threat, we pilot tested each query on the smallest repository and manually verified the results. However, it is still possible that the larger repositories had issues that the small one did not have, or that they characterized the repository entries differently than the smaller repository. After examining the query results, all queries returned valid data and a high-level inspection of the results indicated they were reasonable. Therefore, this threat is minimal.

We also chose projects with greatly differing internal governance and structure. This causes a threat to validity in that there may be differences between the projects caused by the differing internal structure rather than due to the types of coders. However, projects with differing structures had similar results, and the outlying projects had varying structures. Hence, project structure was not a significant threat.

Another threat is that we may have mischaracterized some OTCs. For example, an OTC may have contributed several

times under different user names. Due to the large number of data points in the sample, misidentification of a small number of OTCs should not have effect on the overall results. If a core or peripheral contributor made a single accepted commit under another username, only that particular OTC would be invalid, and thus would not significantly alter the statistical evaluation. Furthermore, by using a different user name for that commit, it is likely that other members of the project would have treated them as any other newcomer. Similarly, incorrectly identifying peripheral or core members should not have altered the data much. The groups of core and peripheral contributors are large enough that misidentification of a small number would not have a significant effect on the overall results. Overall, we are confident in the way we characterized each group of contributors and that any mistakes did not have any systematic impact on the results.

Another mischaracterization threat arises from our measure for core members. Choosing percentage of overall code contributed is a non-traditional approach, and may not accurately capture the core. However, there was a clear cutoff at the chosen percentage between groups of coders, which leads us to believe that it was an acceptable choice for this study.

C. External Threats

Although we did our best to make the study generalizable through mining 21 projects, these projects may not be representative of all FLOSS projects. For example, all of our included projects use the Gerrit code review tool and are large, mature projects. Therefore, this study may not be generalizable to smaller projects or to projects that do not enforce the use of the Gerrit code review tool. However, there is nothing specific about the Gerrit tool that impacted the results. Therefore, FLOSS projects that used other code review tools that captured similar metrics, should see similar results. The results may not generalize to smaller projects, primarily because those projects are unlikely to have a large enough set of OTCs.

IX. CONCLUSION

Core, periphery, and OTCs make up distinct groups in FLOSS projects. Core contributors are usually treated the best, while peripheral and OTC contributors may face obstacles in their attempts to contribute. OTCs face the most obstacles of all, with longer intervals for acceptance, lower acceptance rates, and more involved patch reviews. This validates the data we found in our survey. OTCs do face unique barriers and stronger barriers than other peripheral contributors do; their patches are also different from other peripheral developers.

Using this knowledge, we can work to assist OTCs and other fringe developers in the future by lowering these barriers and finding ways to integrate them into the periphery faster. We can also investigate projects that differed from the trend, such as typo3, where OTCs had the shortest interval, or couchbase, where OTCs had larger patches, in order to determine what these projects are doing differently and potentially understand how to practically lower these barriers for other projects.

ACKNOWLEDGMENT

We acknowledge support from the US National Science Foundation grant number 1322276. We also thank Amiangshu Bosu for creating the original repositories and support in mining those repositories.

REFERENCES

- [1] A. Bosu, J. Carver, R. Guadagno, B. Bassett, D. McCallum, and L. Hochstein. Peer impressions in open source organizations: A survey. *Journal of Systems and Software*, 94:4 – 15, 2014.
- [2] A. Bosu and J. C. Carver. Impact of developer reputation on code review outcomes in oss projects: An empirical investigation. In *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM '14*, pages 33:1–33:10, New York, NY, USA, 2014. ACM.
- [3] N. Ducheneaut. Socialization in an open source software community: A socio-technical analysis. *Comput. Supported Coop. Work*, 14(4):323–368, Aug. 2005.
- [4] K. Lakhani and R. Wolf. *Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects*. MIT Press, Cambridge, 2005.
- [5] A. Lee, J. C. Carver, and A. Bosu. Understanding one time contributors to floss projects. In *39th International Conference on Software Engineering (ICSE)*, 2017.
- [6] S. Panichella. Supporting newcomers in software development projects. In *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 586–589, Sept 2015.
- [7] R. Pham, L. Singer, O. Liskin, F. F. Filho, and K. Schneider. Creating a shared understanding of testing culture on a social coding site. In *2013 35th International Conference on Software Engineering (ICSE)*, pages 112–121, May 2013.
- [8] G. Pinto, I. Steinmacher, and M. A. Gerosa. More common than you think: An in-depth study of casual contributors. In *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, volume 1, pages 112–123, March 2016.
- [9] P. Setia, B. Rajagopalan, V. Sambamurthy, and R. Calantone. How peripheral developers contribute to open-source software development. *Info. Sys. Research*, 23(1):144–163, Mar. 2012.
- [10] S. K. Shah. Motivation, governance, and the viability of hybrid forms in open source software development. *Management Science*, 52(7):1000–1014, 2006.
- [11] I. Steinmacher, T. Conte, M. A. Gerosa, and D. Redmiles. 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*, pages 1379–1392, New York, NY, USA, 2015. ACM.
- [12] I. Steinmacher, I. S. Wiese, T. Conte, M. A. Gerosa, and D. Redmiles. The hard life of open source software project newcomers. In *Proceedings of the 7th International Workshop on Cooperative and Human Aspects of Software Engineering, CHASE 2014*, pages 72–78, New York, NY, USA, 2014. ACM.
- [13] G. von Krogh, S. Spaeth, and K. R. Lakhani. Community, joining, and specialization in open source software innovation: a case study. *Research Policy*, 32(7):1217 – 1241, 2003. Open Source Software Development.
- [14] P. Weissgerber, D. Neu, and S. Diehl. Small patches get in! In *Proceedings of the 2008 International Working Conference on Mining Software Repositories, MSR '08*, pages 67–76, New York, NY, USA, 2008. ACM.
- [15] M. Zhou and A. Mockus. What make long term contributors: Willingness and opportunity in oss community. In *2012 34th International Conference on Software Engineering (ICSE)*, pages 518–528, June 2012.
- [16] M. Zhou and A. Mockus. Who will stay in the floss community? modeling participant’s initial behavior. *IEEE Transactions on Software Engineering*, 41(1):82–99, Jan 2015.