

HKUST SPD - INSTITUTIONAL REPOSITORY

Title	Exploring how software developers work with mention bot in GitHub
Authors	Peng, Zhenhui; Ma, Xiaojuan
Source	CCF Transactions on Pervasive Computing and Interaction, v. 1, September 2019, p. 190-203
Version	Accepted Version
DOI	10.1007/s42486-019-00013-2
Publisher	Springer
Copyright	© the Authors

This version is available at HKUST SPD - Institutional Repository (<https://repository.ust.hk>)

If it is the author's pre-published version, changes introduced as a result of publishing processes such as copy-editing and formatting may not be reflected in this document. For a definitive version of this work, please refer to the published version.

Exploring How Software Developers Work with Mention Bot in GitHub

Zhenhui Peng · Xiaojuan Ma

Received: date / Accepted: date

Abstract Recently, major software development platforms have started to provide automatic reviewer recommendation (ARR) services for pull requests to improve collaborative coding review process. However, the user experience of ARR is under-investigated. In this paper, we use a two-stage mixed-methods approach to study how software developers perceive and work with the Facebook mention bot, one of the most popular ARR bots in GitHub. Specifically, in Stage I, we conduct archival analysis on projects employing mention bot and a user survey to investigate bot performance. A year later, in Stage II, we revisit these projects and conduct additional surveys and interviews with three user groups: project owners, contributors and reviewers. Results show that developers appreciate mention bot saving their efforts, but are bothered by its unstable setting and unbalanced workload allocation. We conclude with design considerations for improving ARR services.

Keywords Automatic reviewer recommendation services · Mixed-methods · User experience · Software development platform

1 Introduction

More and more developers work collectively on software development projects in online platforms such as GitHub. When contributors of a project push their changes of the code to the project repository, pull requests (PRs) are issued and they must be reviewed and approved before the new code gets merged into the codebase (Fig. 1). To maintain the flow of project development, there

Zhenhui Peng
Hong Kong University of Science and Technology
Tel.: +852-62280396
E-mail: zpengab@connect.ust.hk

Xiaojuan Ma
Hong Kong University of Science and Technology

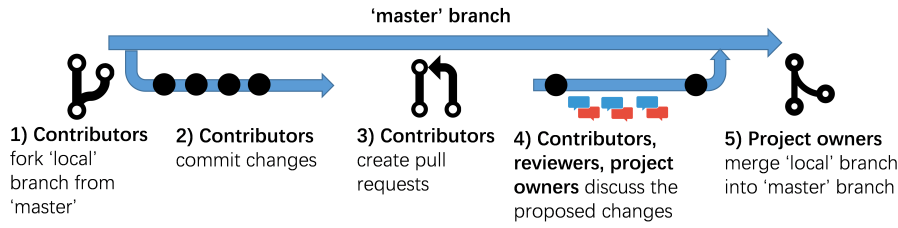


Fig. 1 The pull request process [8].

is a pressing demand for timely, qualified PR reviews [29]. According to a survey in 2014, 15% of the contributors complain that their pull requests hardly get a prompt feedback [13]. It has always been a challenge assigning pull requests to appropriate reviewers [3]. On the one hand, although contributors can propose reviewers in their PRs, many of them, especially those new to a project, have little idea of who may be qualified and willing to review their code [27]. On the other hand, reviewers have limited time and capacity to handle the large quantity of PRs [15,19]. To improve the efficiency of collaborative code review process, some developers introduce bots that provide automatic reviewer recommendation (ARR) service into their projects. For instance, Balachandran [3] implements “ReviewBot” that shortlists potential reviewers by code change history, and then selects the most appropriate reviewer using code review history. Recently, major software development platforms start to provide their own ARR services for users. For example, developers in GitHub with write access to a project’s repository can request reviews from suggested developers based on git blame data [11].

Existing research on ARR services mainly focus on the performance in terms of recommendation accuracy [17,29,30,6]. However, the user experience of and interaction among different stakeholders involved in the process are under-investigated (Fig. 1). In particular, little work has looked into 1) *how developers perceive and work with ARR in practice*, and 2) *what are the most critical needs for different types of users involved in ARR services*. To fill this gap, we conduct a case study on Facebook mention bot, an ARR bot active in GitHub from October 2015 to April 2018, as a lens to gain insights into a better design of ARR services in collaborative software development platforms.

In this paper, we use a two-stage mixed-methods approach to address the two questions mentioned above. In Stage I (2015.11 to 2016.06), we conduct archival analysis on 205 GitHub projects that employed Facebook mention bot at that time. More specifically, for each project, we compare the response rate and response time of pull requests with and without reviewers suggested by mention bot, to assess the effectiveness and efficiency of this ARR bot in practice. We further analyze comments related to mention bot inside these projects and conduct a survey with 52 mention bot users to explore user needs. In Stage II (2016.07 to 2017.08), we revisit these projects and analyze

new user comments emerged within the year to see if user needs identified in Stage I are met. To gain more in-depth understanding of why developers use / do not use mention bot and what they expect from an ARR service, we divide the ARR users into three groups: project owners, contributors, and reviewers. Then we conduct an additional survey with thirty-six valid responses and interview six developers in GitHub to explore the needs of each group. Results of the two-stage investigation show that developers appreciate mention bot saving their efforts, but different user groups have different demands for ARR services, i.e., simplicity and stability needed by project owners, transparency needed by contributors, while selectivity needed by reviewers. We summarize our findings into considerations for future ARR services design.

2 Background and Related Work

In this section, we first introduce the concepts of pull request (PR) and review process in GitHub, then we summarize the mechanism of some existing ARR services.

2.1 Pull request and Review Process in GitHub

The pull-based development is the latest model of distributed software development [12]. To receive external contributions, repositories are shared by fork (i.e., clone) and modified by PRs. Normally there are three kinds of developers involving in the pull request process:

- Project owners who possess PRs in their projects.
- Contributors who submit PRs that need reviews.
- Reviewers who help to review PRs.

The pull request process is described in Fig. 1. Contributors fork a master branch and commit changes to their local branches [12,14,15]. To make contributions to the master branch, contributors submit a set of changes by creating a PR. The owners inspect the PR and project owners decide whether to merge the changes or not. During this process, the project owners, reviewers and contributors usually need to discuss the proposed changes. In the end, the PR is closed.

After a pull request is opened, anyone with read access can review and comment on the changes it proposes. GitHub allows developers to comment on the changes proposed in pull requests, approve the changes, or request further changes before the pull request is merged. When PRs are submitted, they are intended to be reviewed within a short period of time. However, in reality, owners in popular projects receive too many PRs. They have difficulties in reviewing these PRs by themselves or identifying other appropriate reviewers for them [3,15,17,24,25,27,28,29,30].

2.2 Automatic Reviewer Recommendation for Pull Requests

To reduce project owners' efforts, some researchers have proposed automatic reviewer recommendation (ARR) services [3,17,24,27,29,30]. As the key of any review is context and change understanding [1], these ARR services intend to bring in reviewers who are qualified for the PRs and willing to help. They normally use historical information of code change and review in order to identify appropriate reviewers [3,17,24,27,29,30]. The "ReviewBot" proposed by Balachandran shortlists potential reviewers by blame information, and then selects the most appropriate reviewer who has modified the related code sections most [3]. Thongtanunam et al. proposed "RevFinder" which recommends reviewers not only based on code review history but also the similarity of file paths [24]. Then, "Tie" was proposed to enhance "RevFinder" by using different similarity measures for file paths and textual information in pull requests [27]. Jiang et al. developed "CoreDevRec" to train a prediction model using a support vector machine [17]. This model uses three features, which are file path, social interaction between reviewers and contributors, and activeness of reviewers. Profiles of the developers are also used for reviewer recommendations. For example, Rahman et al. proposed to use the experience of a developer in certain specialized technologies associated with a PR in addition to the cross-project experience to determine the expertise as a potential code reviewer [20]. The experiment on their dataset show that this technique can achieve over 85% recommendation accuracy. Fejzer et al. employed a similarity function between programmers' profiles and change proposals to be reviewed to give recommendations, and they obtained improved results in terms of classification metrics and performance [6]. A review of different PR reviewer recommendation techniques can be found in [2].

While the above ARR services are outside tools of the software development platforms, GitHub has provided their own ARR services. The "CODE-OWNERS file" [10] is used to define individuals or teams that are responsible for the code in a repository. These developers will be automatically requested for review if someone modifies the code they own. The "suggested reviewer" feature [11] can automatically suggest reviewers based on git blame data. Every time a PR is submitted, the organization members, repository owners and collaborators can see the suggested reviewers in the right sidebar of the PR and they can decide whether to request reviews from these reviewers or not.

However, little work addresses how software developers perceive and work with these ARR services in practice. Many factors could affect the efficiency of these services, as suggested by works that explore user experience of recommendation services in the domains of music, digital cameras [4,7,18,22,23]. For example, Sinha et al. [22] studied the role of transparency in music recommender systems. Stolze et al. [23] found that compared with the feature-oriented recommendation, needs-oriented recommendation for digital cameras was more helpful. Chen Li et al. [4] studied how personality influences users' need for recommendation diversity. Ferwerda et al. [7] tested

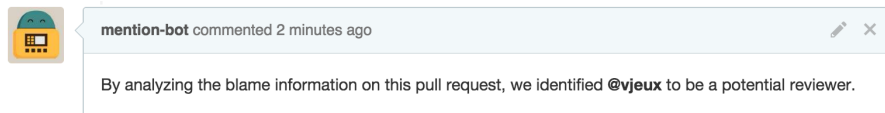


Fig. 2 An example of the mention bot comments [5].

that the user personality affected their ways of choosing music. However, the methods to study the user experience in above scenarios only referred to only one specific aspect or no more than two user groups. It is still a challenge to study user experience in the domain of ARR service for online software development platforms, which might involve three kinds of user groups (project owner, contributor and reviewer) in.

2.3 Facebook Mention Bot

Facebook mention bot can recommend any developers to be reviewers using two heuristics: 1) If a line was deleted or modified, the person that last touched that line is likely to care about this pull request. 2) If a person last touched many lines in the file where the change was made, he may want to be notified [5].

Since its launch in October 2015, mention bot had served for 205 GitHub projects and handled 12060 pull requests up to June 2016. owners of GitHub projects can deploy the mention bot using a webhook service [26] without any extra setting. Once the mention bot is employed in a project, a recommendation comment is added to the newly made pull requests as shown in Fig. 2. By default, mention bot will straightly mention its recommended reviewers after the PR is created, but project owners can manually personalize the bot by adding a “.mention-bot” file to the base directory of the repository [5]. For instance, they can configure some recommendation and notification rules such as the maximum number of candidates for recommendations, the message from mention bot and the blacklist for some reviewers.

3 Research Method Overview

In this section, we first introduce the facebook mention bot, and then present our two-stage mixed-methods approach.

3.1 Facebook Mention Bot

In this work, we use mention bot developed by facebook as a lens to look into how developers work with it in practice and what are the critical needs for different stakeholders. Facebook mention bot can recommend any developers

to be reviewers using two heuristics: 1) If a line was deleted or modified, the person that last touched that line is likely to care about this pull request. 2) If a person last touched many lines in the file where the change was made, he may want to be notified [5].

Since its launch in October 2015, mention-bot had served for 205 GitHub projects and handled 12060 pull requests up to June 2016. owners of GitHub projects can deploy the mention-bot using a webhook service [26] without any extra setting. Once the mention-bot is employed in a project, a recommendation comment is added to the newly made pull requests as shown in Fig. 2. By default, mention-bot will straightly mention its recommended reviewers after the PR is created, but project owners can manually personalize the bot by adding a “.mention-bot” file to the base directory of the repository [5]. For instance, they can configure some recommendation and notification rules such as the maximum number of candidates for recommendations, the message from mention bot and the blacklist for some reviewers.

3.2 Two-stage Mixed-methods Approach

To better explore how software developers perceive and work with Facebook mention-bot overtime, we carry out our research with archival data, survey and interview in two stages. In the first stage, we analyze 205 projects that employ mention bot, investigate 53 issue comments about mention-bot, and conduct a survey with 52 mention-bot users. In this stage, we focus on mention bot’s performance in practice during a certain period (from November 2015 to June 2016). By analyzing the pull requests in these projects, we measure the response rate and the response time of the recommended reviewers. We use the issue comments to investigate user needs for mention-bot, while the survey is used to learn how users perceive its usefulness. We conclude with three potential features to improve mention bot and address user needs at the end of Stage I. In the second stage, we revisit these projects and analyze another 90 related comments emerged within this year to see if user needs identified in Stage I are met. Furthermore, to gain more in-depth understanding of why people use/do not use mention bot and what they expect from an ARR service, we conduct a survey and acquire 34 valid responses from three user groups, i.e., project owners, contributors and reviewers, and then interview six developers. Then we explore factors critical to the user experience of ARR services for each user group. Noticed that above research methods might conflict with or support each other, we then integrate our results of two stages to discuss how to provide better user experience in automatic reviewer recommendation services.





Features	1st quartile	Median	Mean	3rd quartile	Histogram
Development period (months)	9.5	20.63	27.8	43.6	
Size (SLOC)	1.5K	9.37K	58.01K	41.1K	
Commits	188	546	2,863.48	2,855	
Pull requests	21	69	413.30	246	

Table 1 Properties of projects used in our archival analysis

Likability	# of issue comments
Positive	25
Negative	20
Neural	8

Table 2 The number of issues comments that show the positive, negative and neural evaluations of the mention bot.

4 Stage I

4.1 Archival Data and Survey Collection

In Stage I, we track the public activities of Facebook mention bot up to June 2016 and identify 205 projects in GitHub that employ this bot. We use GitHub API [9] to gather their properties, pull requests and issues. Among these projects, we exclude the projects that have less than four reviewer candidates (i.e., the total number of contributors in the project), since the mention bot normally recommends up to three candidates. We further exclude the projects that have not received any external contribution (i.e., pull requests made by external contributors). According to the literature, a reviewer identification task can be challenged with external contributions [15,25,28]. Finally, we use 155 projects for our investigation. We exclude the following pull requests: 1) Pull requests made by project owners and merged into a master branch without any review. 2) Pull requests made by other bots. 3) Pull requests not closed. In total, we identify 64,937 pull requests from the 155 projects. Among them, the mention bot is called in 9,413 pull requests while not being used in the rest.

Table 1 represents the properties of the 155 projects in our dataset up to June 30th, 2016. Their average development period is about 28 months ($SD = 21.38$). The latest revisions of the projects have approximately 58K lines of source code on average excluding whitespace and comments ($SD = 124.71K$). The average numbers of commits and pull requests in total are around 2,863 ($SD = 7,953.35$) and 413 ($SD = 1,250.60$) respectively for each project.

We extract 258 issue comments that contain the keyword, “mention bot” in the original 205 projects. To avoid bias, we exclude 15 issue comments from the two projects that develop and test the mention bot. Through manual inspection, we finally identify 53 issue comments that express the likeability

of the mention bot (Table 2). There are 25 positive comments, 20 negative comments and eight neutral comments that give suggestions.

In Stage I, we identify 2,467 developers in GitHub who make or review the pull requests that call Facebook mention bot. Among them, 1,445 developers post their email addresses on GitHub profiles or personal web pages. We advertise for our survey to these developers by emails. To get more responses, we also invite them to distribute the survey to their communities. In total, we receive 52 responses.

Our survey consists of five questions about the perceived usefulness and likeability of Facebook mention bot. The first question asks if mention bot recommendations are appropriate. We use a 5-point Likert scale to measure the appropriateness of the mention bot recommendations. In the next question, we measure the perceived reduction in response time and efforts after deploying the mention bot. We provide four statements regarding this aspect and ask the respondents about their level of agreement using a 5-point Likert scale. The first two statements represent whether participants receive responses faster or provide a faster response when the mention bot is involved. The other two statements are to examine whether the participants can save the efforts spent on identifying proper reviewers or exploring pull requests using the mention bot. The rest three questions ask about the likeability of the mention bot. Specifically, we use a 5-point Likert scale to measure how much the participants like the mention bot. Then, we offer the four options that correspond to the “Reviewer recommendation”, “Automatic notification”, “Enable/disable notification for certain PRs/people” and “Message customization” features of the mention bot. We ask the participants to select one or multiple favorite features if they respond positively to the previous question. Finally, a yes-no question asks if they would continue using the mention bot.

4.2 Analysis and Findings

4.2.1 Performance of mention bot

To understand what kind of benefits a reviewer recommendation service can provide, we first technically measure mention bot’s performance by response rate and response time of recommended reviewers in practice. In our work, we measure response rate rather than top- k accuracy as in other works [3, 17, 24, 27] because contributors concern about whether there is any response from recommended reviewers. If the ARR service can correctly recommend a reviewer who is interested in working on the PR even he or she might not work it out, it still does a good job. Response rate (Equation 1) represents the percentage of pull requests whose actual reviewers are correctly recommended by Facebook mention bot. It is similar to top- k accuracy, but we count it as a hit if any of the recommended developers is observed in a review process.

$$\text{Response rate} = \frac{\sum_{r \in R} \text{Hit}(r, \text{Response})}{|R|} \times 100\% \quad (1)$$

Response time difference	# of projects
$PR_{Non-bot} > PR_{Bot}$ (p-value < 0.05)	25
$PR_{Non-bot} < PR_{Bot}$ (p-value < 0.05)	6
No significant difference	124

Table 3 We compared the response time in the PR_{Bot} and $PR_{Non-bot}$ groups using Mann-Whitney-Wilcoxon test.

The calculation of response rate for each project is straightforward. For each pull request (PR) that mention bot comments on, we count it as a successful response if at least one of the recommended reviewers show up in this PR review process. Overall the average of the response rate in the 155 projects is about 75.37% (SD = 26.92%).

Response time (Equation 2) can reflect whether mention bot can reduce time in involving reviewers in pull requests. It refers to the time difference between submitting a PR and the first response made by any developer other than the submitter [28,29].

$$\text{Response time} = T_{FirstResponse} - T_{SubmitPR} \quad (2)$$

To measure the response time of recommended reviewers, for each project, we divide the pull requests into the two groups by whether the mention bot is called, and compare the average response time between the two groups. In detail, we put the pull requests that call the mention bot into the PR_{Bot} group and the rest of them into the $PR_{Non-bot}$ group. After excluding the responses made by bots, we calculate the average of the response time in each group. To precisely calculate the average, we exclude the outliers using the interquartile range (IQR) in Box-and-Whisker plots [16]. We found that the response time is reduced in 75 out of the 155 projects (about 48.4%) when deploying the mention bot. However, the response time rather increase in the rest of the projects.

We further analyze the change in the response time using Mann-Whitney-Wilcoxon test. Table 3 shows the comparison between the response time in the PR_{Bot} and $PR_{Non-bot}$ groups. When the mention bot is deployed, the response time in the 25 projects is significantly reduced while the response time in the 6 projects is significantly increased. In the rest of the projects, there is no significant time difference between the two groups. We then randomly sample pull requests that do not employ the mention bot from the 6 projects whose response time significantly increased ($PR_{Non-bot} < PR_{Bot}$). The average response time is about 1.7 hours (SD = 5.14). However, in the 25 projects with a significant decrease in response time ($PR_{Non-bot} > PR_{Bot}$), the average response time is around 9.45 hours (SD = 74.01). This result implies that mention bot is more likely to reduce the response time in less active projects.

In our survey, we evaluate developers' perceived usefulness and likeability of mention bot. Fig. 3(a) shows the survey results of the first question that

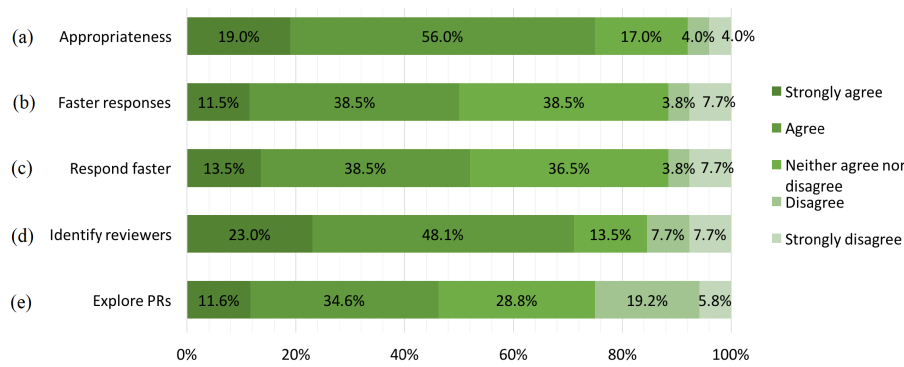


Fig. 3 Participants indicate their level of agreement with following statements: (a) I think the recommendations made by the mention bot are appropriate. (b) I receive faster responses from reviewers using mention bot. (c) I respond faster to review request sent through the mention bot. (d) Using the mention bot saves my efforts to identify proper reviewers. (e) Using the mention bot saves my efforts to explore PRs.

asks the appropriateness of Facebook mention bot recommendations. About 75% of the participants express positive responses with strongly agree or agree. In the second question, the first two statements ask whether the participants could save time when using the mention bot. As shown in Fig. 3(b)(c), 50-52% of the participants strongly agree or agree with the statements while 36.5-38.5% of the participants neither agree nor disagree with them. The rest 11.5% of the participants strongly disagree or disagree with the time benefit from the mention bot. The last two statements in the second question ask if the participants could reduce efforts with the mention bot. Overall, compared to the responses for the time reduction, there are more positive and negative responses but less neutral responses. As described in Fig 3(d)(e), 46.2-71.1% of the participants give us positive responses (strongly agree or agree) while 13.5-28.8% of them reply with the neutral (neither agree nor disagree). 15.4-25% of the participants show the negative responses (strongly disagree or disagree).

Interestingly, about 20% of the participants respond that the mention bot is useful to save the efforts for identifying proper reviewers but not helpful to reduce the time spent in this process. These results may imply that the effort reduction in identifying reviewers is perceived as the key benefit that mention bot provides for developers.

In the survey, we ask the participants whether they like mention bot and whether they would continue using it. The results show that 73% of the participants strongly like or like the service and 84.6% of the participants would continue using it, which suggests that users are positive about mention bot.

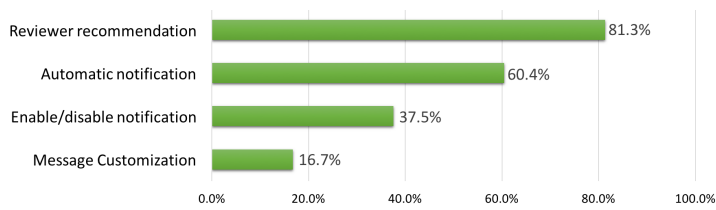


Fig. 4 The favorite features of the mention bot. The participants can choose one or multiple features.

4.2.2 User needs for mention bot

In our survey, we ask the participants to indicate their level of agreements with mention bot's features: "Reviewer recommendation", "Automatic notification", "Enable/disable notification for certain PRs/people" and "Message customization". Fig. 4 shows the result from the survey. The most favorite feature is the "Reviewer recommendation" with the support from about 81.3% of the participants. The second most favorite feature is the "Automatic notification" which receives votes from approximately 60.4% of the respondents. The other two features, "Enable/disable notification for PRs/people" and "Message customization", are the favorites for around 37.5% and 16.7% of the participants, respectively. The 53 issue comments also show users' preference about these features. As showed in Table 2, 25 issue comments contain positive feedbacks on the mention bot. The developers seem to like its core features, including the "Reviewer recommendation" and "Automatic notification". Especially, when the mention bot is shut down [21], we observe developers feel inconvenient and manually send notifications to the potential reviewers:

"@YYY could you take a look at this and #2645 if you have time [...] Not sure what happened to our friend the mentionbot. facebook/mention-bot#134"

However, in the 20 comments of negative feedbacks on the mention bot, developers dislike the mention bot's insensitivity to context and unbalanced workload allocation. Some of them do not want to get further notifications because they no longer work on the projects:

"Can someone please correct the blacklist for @mention-bot? I don't want to receive any notifications for this repository as I'm not a collaborator here. PS: Just complaining because this is the 4th email I receive thanks to the bot."

While the context insensitivity problem bothers the developers who no longer work on the projects, the unbalanced workload allocation problem increases some reviewers' workloads and discourage others:

"If a person is being recommended a lot, nominate a reviewer who wouldn't have a super hard time"

"It's almost always recommending the same person in our project which is not really that helpful."

The main cause of these problems lies on its manual setting. In the current environment, project owners have to manually identify developers who do not want to be notified and then add them to the blacklist. The “Enable/disable notification for certain PRs/people” feature of mention bot is designed to minimize the above incorrect recommendation and unbalanced workload allocation problems, but its unfriendly designation discourages the users (only 37.5% of the participants like it).

The results from the survey and comment analysis imply that “Reviewer recommendation” and “Automatic notification” are the key features of mention bot (favorite by 83.1% and 60.4%, respectively). When these two features are broken, users will feel inconvenient. But if mention bot keeps notifying a specific reviewer, it will increase the workload of the reviewer. And users need a higher context sensitivity which can avoid notification to inactive developers in the projects. Given with these results, we find the possibility that the user needs may come from three user groups. For example, the project owners and contributors need the “Reviewer recommendation” and “Automatic notification” features, while the reviewers need a more balanced workload allocation and a higher context sensitivity.

4.2.3 Potential Features to Improve Mention Bot

To address above user needs, we propose three potential features to improve mention bot:

- A delay for 1-3 days before activate mention bot
User comments suggest that the immediate activation of mention bot may cause redundant notifications to developers. We explore the distribution of the response time in the archival data. We find that about 80.34% and 89.52% of the pull requests are responded within 24 and 72 hours respectively. Given this, we propose that a delay for 1-3 days before activate mention bot would help to avoid the majority of redundant notification.
- Automatically disable notification for inactive developers
We find that the notification feature may bother developers who no longer work on the projects. Mention bot does have a blacklist to not notify certain developers, but project owners need to manually identify these developers and add them to the blacklist. We propose a feature that automatically turns off notifications if reviewer candidates are inactive. We suggest measuring the activeness of developers by checking their last contribution to the project and how much times they fail to respond to the recommendations before. For example, if a reviewer candidate made the last contribution on a project six months ago and fails to respond to the notifications three times, it would be better to turn off the notification to this developer.
- Limit the maximum number of review requests to one developer.
Workload balancing among recommended reviewers can be critical. As the participants say, it is not realistic to ask one developer to review many pull requests at a time while others have nothing to work with. We propose that we can limit the maximum number of review requests to one

developer. For example, if one developer receives more than five review requests within a week, it is reasonable to lower the priority of this developer in recommendations.

Overall, up to June 2016, Facebook mention bot performs quite well as a reviewer recommendation service. Its recommended reviewers respond actively to the PRs (75.37%) and it is useful to reduce the response time in less active projects. Our user study supports that mention bot recommends appropriate reviewers for the PRs (75%) and developers perceive that the effort reduction in identifying reviewers is the key benefit provided by mention bot. And we find the possibility that the user needs identified may come from different user groups, which motivates us to investigate factors critical to their experience of ARR services separately in the later stage.

5 Stage II

Over the year, Facebook mention bot has added more configuration options to benefit different users. For example, the “delayed” feature that we proposed in Stage I is added with default “false” setting and a “delayUntil 3 days” configuration. In addition, to avoid redundant notifications to the reviewers and provide a better recommendation result for the contributors, project owners can now filter developers and files via settings such as “requiredOrgs”, “skipAlreadyMentionedPR”, “fileBlacklist” and “skipTitle”. With so many attractive features added in, it is interesting to know *whether mention bot has attracted more projects, whether developers are satisfied with the improvement, and whether each of the three user groups have unmet needs and expectations for ARR service*. To answer these questions, we conduct the second stage of our research, starting with re-analyzing the adoption of mention bot and investigating factors critical to the three user groups of ARR services, i.e., project owners, contributors and reviewers, respectively.

5.1 Re-analysis of the Adoption of Mention Bot

5.1.1 Archival analysis

In Stage II, we find that the official account of “mention-bot” has been removed from GitHub so that we can not track the mention bot’s activities in the PRs like we do in Stage I anymore. But mention bot is still active. On the one hand, reviewers are still notified by mention bot. On the other hand, some developers configure mention bot by adding a “.mention-bot” file to the base directory of the repository and use different accounts to comment on the PRs, such as “jimmibot” in “syndesisio / syndesis-ui” repository and “salt-jenkins” in “saltstack/salt” repository. Therefore, we revisit the 205 projects to check whether they still use mention bot using following criteria: 1) Removed: Some issues explicitly claim that the project removes mention bot.

Status	Removed	Still use	Disappeared	Unclear	Newly add
# of projects	22	30	11	142	22

Table 4 Mention bot’s status in the projects in Stage II

Contents	#	Details
Benefits	5	Automatic notification; reviewer recommendation; involve more reviews in
Workload allocation	9	The same people; aggressive notification; want to be added in blacklist
Bug	12	Configuration problem; ignore the config. file; not active
Suggestions	9	Turn it into plug-in; recommend experts; whitelist; provide some links
Alternative	8	GitHub “suggested reviewers”; CODEOWNERS; dwylbot

Table 5 Contents showed in some comments

2) Still use: The “.mention-bot” configuration file still exists in the project or there are issues that imply the existence of mention bot. 3) Disappeared: The project no longer exists in GitHub. 4) Unclear: There is no “.mention-bot” configuration file inside the project and we cannot find any issues claiming that the mention bot is still use or has been removed.

Through our manual check, we find that 22 projects have removed mention bot, 30 projects still use it, 11 projects disappear and the rest 142 projects are unclear (Table 4). Among these 142 projects, 72 of them have no issues about mention bot, which is unusual if mention bot serves well in these projects. In addition, we try to identify mention-bot-related activities in other GitHub projects between July 2016 and August 2017 by searching “create mention bot” and “remove mention bot” in project “Commits” log in GitHub. Filtering out the irrelevant results, we identify that 22 projects claim that they employ mention bot and 19 projects claim that they remove it during this period. We can see that mention bot is not increasingly used in GitHub during this year.

To collect users comments about mention bot, we search related issues in GitHub using the keyword “mention bot”. Filtering out the irrelevant results, we finally identify 90 effective comments emerged within the year (from July 2016 to August 2017) that express user attitude towards mention bot. Among these comments, 33 of them show positive attitude toward mention bot, 26 of them are negative, and the rest 31 are neural. We further classify these comments based on their contents (Table 5), and identify five comments specifying benefits of mention bot, nine complain about the unbalanced workload allocation, 12 comments reporting bugs, nine suggesting room for improvement, and eight proposing an alternative service. We suspect that the adoption of mention bot is greatly impaired by its bugs, unbalanced workload allocation problem and the existence of alternatives.

5.1.2 Survey and Interview

To further investigate developers’ perceived usefulness of mention bot and explore factors critical to ARR user experiences and adoption, we conduct a

survey with three user groups: project owners, contributors and reviewers (see Research Method Overview Section). We design five different questionnaires in our survey: 1) Project owner using mention bot; 2) Project owner not using mention bot; 3) Contributor using mention bot; 4) Reviewer using mention bot; 5) Contributor or reviewer not using mention bot. We design only one questionnaire for the contributor not using mention bot and the reviewer not using mention bot because we ask almost the same questions to investigate their needs and expectation for ARR services. Across all user groups, we ask respondents to rate the perceived usefulness and annoyance of the service as well as the efficacy of each features of mention bot on a 5-point Likert scale (1 being the least of each measure). There are also customized questions for each user group. For example, we ask project owners about the reason why they (do not) deploy mention bot; we ask contributors what they would do before issuing a pull request and when a mention bot comments on their pull requests; and we ask reviewers how they get pull requests to review and what they would do if notified by mention bot.

By searching the contributors of the projects that use mention bot now or used it before, we sent emails to over 700 potential users for invitation to survey and interview. Noticed that software developers might act as project owners, contributors or reviewers under different circumstances, we ask the participants to fill out the surveys as much as they could if they match the criteria and invite them to join our interview. In total, we get 34 effective survey responses and interview six developers through email and Google hangout.

We receive a total 34 valid responses from our survey. The responses come from 7 project owners using mention bot, 10 project owners not using mention bot, 11 contributors or reviewers not using mention bot, five contributors and one reviewer using mention bot. Overall, mention bot users “find it useful” (mean = 4.08, SD = 0.64). The project owners employ mention bot in their projects mostly for its “efficiency” (mean = 4.29, SD = 0.95) or “convenience” (mean = 3.71, SD = 0.95), but not for “fun” (mean = 2.29, SD = 1.11). With mention bot, project owners spend less effort in “managing the pull request process” (mean = 3.71, SD = 0.76) and can “engage developers more in the projects” (mean = 3.86, SD = 0.69). However, employing mention bot does not necessarily “boost the activeness of the projects” (mean = 3.00, SD = 1.00). After briefly explaining the concept of mention bot to the contributors and reviewers who have never heard of the service, 70% of them hope that the projects they participate in would employ it. Contributors who use mention bot do not think that they can always “get faster response from its recommended reviewers than from others” (mean = 3.00, SD = 0.71), or that “the suggested reviewers certainly provide better feedback” (mean = 3.20, SD = 1.10), or that “it improves their interaction with other developers” (mean = 3.2, SD = 1.10). However, they do agree that it saves their efforts in looking proper reviewers (mean = 4, SD = 1.22), which is consistent with our findings in Stage I.

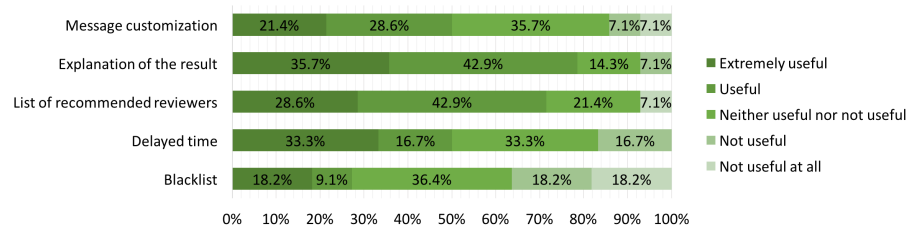


Fig. 5 The potential features of a reviewer recommendation service. Participants are asked to evaluate their usefulness.

Among the respondents, six software developers (I1,2,3,4,5,6) express further interest and join our semi-structural online interview. I1 is a project owner as well as a current user of mention bot and I2 is a reviewer as well as a contributor who did not hear about mention bot before and the rest four (I3,4,5,6) are project owners who used it before but removed it later. We mainly ask them to share their user experience with or without mention bot during the interviews.

When we ask I3,4,5,6 why they removed mention bot, surprisingly, their answers are quite similar:

“We’re not using mention bot any longer because GitHub added the “suggested reviewers” feature which is enough for our needs, but we found mention bot very useful otherwise.” (I3)

Compared with mention bot, the “suggested reviewers” feature in GitHub is less aggressive because it does not automatically notify the reviewers but only suggests potential reviewers to project owners who have the write access to the PRs. It is plugged into the GitHub platform so that users do not need to configure it by themselves and worry about its instability. However, interviewees also commented that “suggested reviewers” is not flexible enough, as developers who only have read access to the PRs cannot send a request to the suggested reviewers on their own if the project owners are too busy to notify them. Our interviewee I1, the owner of a big project (with 1870 contributors, 84888 commits and 26576 closed PRs up to August 25, 2017), explains why he continues to use mention bot rather than “suggested reviewers”:

“Our project is too big. The feature needs permission, the suggested reviewers should be the member of our project. But we have nearly 2000 contributors. We want them all in our project, and mention bot suits our need.” (I1)

He stresses how mention bot contributes to his project:

“Mention bot does improve the quality of software, because more people review the pull request before they are merged. A big improvement of the number of reviews that we get.”

Overall, we find that mention bot's performance does not meet some developers' expectation possibly because of its instable settings, unbalanced workload allocation and the existence of other ARR services, especially the better integrated "suggested reviewers" feature of GitHub. Still, many users value mention bot's benefits in terms of extending reviewer pool and reducing effort in managing PRs. In the next subsection, we present the factors essential to the unique experience of each ARR user group.

5.2 Factors Critical to ARR User Experiences

In our survey, we ask participants to indicate their perceived usefulness of a list of potential features of a PR reviewer recommendation service identified in Stage I: "Message customization", "Explanation of the result", "List of recommended reviewers", "Delayed time" and "Blacklist". As shown in Fig. 5, most respondents find "Explanation of the result" and "List of recommendation reviewers" (extremely) useful features to have (78.6% and 71.5%, respectively). In comparison, respondents' perception of the other three features which already exist in mention bot is rather neutral (50.0% for "Message customization" and 50.0% for "Delayed time", and 27.3% for "Blacklist"). In fact, some comments from social media are negative about these features:

"how do I get myself blacklisted from this XXX mention bot thing?"

"If the delay feature is enabled, mention bot no longer works"

We further explore features that matter most to project owners, contributors and reviewers.

– Project owners

1) **Simplicity.** Although Facebook mention bot claims that it can be set up easily, it has 22 configuration options now. We find that most of the projects we visit just keep the default setting, which disables features that might be helpful for contributors and reviewers such as *fileBlacklist*, *Skip-Title* and *requiredOrgs*. In fact, some project owners removed mention bot because of they could not configure it right:

"The configuration added is not working, so I just removed it since the benefit would be minor anyways (and might annoy some people?) Very funny. I was deleting the mention bot webhook and accidentally found out why it was not working. I forgot to check the events to be sent on 'Labeling'. Oh well."

2) **Stability.** Mention bot itself is a project under constant development, and thus may not function normally from time to time, which really affect the experience if it is under heavy usage. One of our interviewee (I4) removed it because *"It stopped working a while ago so I've disabled it."* Besides, as showed in Table 5, the bugs of mention bot reported by 12 out of the 90 comments we collected also discourage its usage, e.g., *"Seems the complete .mention-bot file is currently ignored"*.

– Contributors

3) **Transparency.** According to our survey, it is not a common practice for contributors to identify reviewers by their own, such as “manually search and add reviewers” (mean = 2.57, SD = 1.16) or “mention reviewers they know” (mean = 3.00, SD = 1.03). When mention bot comments on their PRs, although they are inclined to “trust its recommendation” (mean = 3.8, SD = 0.84), many contributors will still “check its recommendation” (mean = 4.6, SD = 0.55). But mention bot and GitHub “suggested reviewer” feature are not transparent enough as their results are only several user names of the reviewers. Some contributors may want to know who is in charge of the part that they make PR to: *“Normally I want my direct supervisor rather than those who had modified related files to review my PRs”* (I2). The fact that “Explanation of the result” and “List of recommended reviewers” are the most preferred features according to our survey also suggests that contributors want decisions made by ARR services to be more transparent.

– Reviewers

4) **Selectivity.** Our respondents are somewhat conservative about taking on PR reviews, such as “look for pull requests interesting to me on my own” (mean = 3.33, SD = 1.12), “mentioned by contributors” (mean = 3.00, SD = 1.12) or “want to be recommended by a bot” (mean = 3.33, SD = 1.00). This may be because they are already rather occupied: *“I am too busy to look every email from GitHub because it sends all information about the update of pull requests, but actually I do not need to review all those pull requests”* (I2). I2 said that he would like the bot to only notify him with the PRs that really need him. These results imply that the reviewers would not actively take on ordinary PR reviews but want to have selectivity to only be notified by certain kinds of PRs.

Overall, in Stage I we find that users need a better reviewer recommendation with automatic notification as well as a more balanced workload allocation and a higher context sensitivity, while in Stage II we further explore that the simplicity, stability, transparency and selectivity are critical to the ARR experiences of different user groups. Taking all these user needs and factors into account, we propose our design considerations of ARR services in next section.

6 Discussion

In this section, we present design considerations for improving ARR services, other insights and limitations of this work.

6.1 Design Considerations for Improving ARR Services

Based on findings from both stages, we propose three design considerations that would possibly improve user experience of ARR services.

6.1.1 Easier Configuration of ARR Service for Project Owners

Users cannot customize the “suggested reviewers” feature provided by GitHub, and thus it cannot adequately meet different types of user needs. Mention bot does have many options to deal with different situations, but its unfriendly manual configuration process intimidates many project owners who are responsible for handling the service. Since the project owners tend to “*only care about the PRs and want the bot easily tells how its capacities are*” (I1), we propose that a better ARR service should have an easier configuration process. For example, the service can have shortcuts to easily change modes to satisfy different needs. If the project needs more external contributions, the owner can use a shortcut to adjust some options to invite external reviewers to review the PRs. Besides, the service can have a log so that project owners can easily reset it to the suitable and stable state.

6.1.2 Better Transparency of Recommendation for Contributors

According to our survey in Stage II, contributors tend to check on mention bot’s recommendation when it comments on their PRs, and they call for information that can improve their understanding of why a particular recommendation is made. Therefore, we propose that a better ARR service should keep their recommendation transparent to contributors, especially regarding the qualification and availability of the suggested reviewers. For each PR, in addition to directly naming the top few appropriate reviewers, ARR service can provide a ranked list of all the potential reviewers for this PR, each with a brief profile summarizing their role in the project, specialty, recent activeness, current workload, etc. In case contributors would like to manually select reviewers, this list would be a good place to start.

6.1.3 More Flexible Notification Preference Setting for Reviewers

Reviewers are bothered the most by ignorant PR review notifications. For example, when reviewers are already overloaded with work on the project or in real life, they do not want to receive more review requests. Mention bot does have some mechanisms to filter reviewers in the candidate pool, but only project owners have the access to set the rules. Reviewers have to contact the managers to adjust the pool if they would like to disengage from / reengage in the review activities. While the automatically filtering out inactive reviewers feature that we propose in Stage I is a potential way to avoid unnecessary notification, and it may not be able to respond instantly to urgent changes

in availability. Hence, we propose that a better ARR service should allow reviewers to specify personal notification preference on their side. Reviewers can change their status to “Do not disturb” when occupied, declare types of PRs uninterested to them, and set a maximum quota of PRs. Further more, for reviewers (e.g., I2) who would love to help but are not sure of their qualification and/or availability, ARR services may instead recommend PRs to them according to their interests.

6.2 Additional Insights into Bot Usage in GitHub

Our interviewees share their positive attitude toward general bots usage in GitHub in the interviews.

*“Really necessary, because there are many repeated work to do otherwise.”
(I1) “I feel most of the bots can solve actual problems. I am positive toward these bots and hope more and more useful bots come up.”(I2)*

Almost every big project that we visit for this research involves some bot(s) in its development, such as “facebook-github-bot” in Facebook organization, “Microsoft Pull Request Bot” in Microsoft society and “greenkeeper bot”. Their functions are very specific, helping with small chores like adding labels to PRs or sending customized messages. Our interviewees hope to see a bot that can provide all these functions in the future.

6.3 Limitation and Future Work

Our work has some limitations. Some of our findings might be unique to Github and we did not compare Facebook mention bot with other ARR services. Our survey results come from a small sample of developers due to the low response rate. Therefore, our results may not represent the opinions of the entire user group (e.g., contributors with different levels of experiences) about reviewer recommendation services. In the future, we plan to improve the coverage and generality of our research, develop a user-friendly ARR service based on the findings, and test its usefulness, usability, and user experience in the wild.

7 Conclusion

In this paper, we used Facebook mention bot, an automatic reviewer recommendation (ARR) bot in GitHub, as a lens to explore how developers work with ARR services. We used a two-stage mixed-methods approach to investigate practical usefulness of mention bot and critical needs for different types of users. Our Stage I investigation (June 2016) shows that mention bot performed quite well as it can save contributors’ effort in identifying proper reviewers, and can achieve a 75.57% response rate among suggested reviewers who expressed the need for a better workload allocation. A year later in

Stage II (August 2017), we do not see an obvious increase in mention bot's adoption, perhaps due to its inherent problems and the existence of other ARR alternatives. Our survey and interview with three user groups (project owners, contributors and reviewers) suggest that simplicity, stability, transparency and selectivity are critical to the user experiences of ARR services. According to these findings, we propose a set of considerations for designing more user-friendly ARR services.

References

1. Bacchelli, A., Bird, C.: Expectations, outcomes, and challenges of modern code review. In: Proceedings of the 2013 International Conference on Software Engineering, ICSE '13, pp. 712–721. IEEE Press, Piscataway, NJ, USA (2013). URL <http://dl.acm.org/citation.cfm?id=2486788.2486882>
2. Badampudi, D., Britto, R., Unterkalmsteiner, M.: Modern code reviews - preliminary results of a systematic mapping study. In: Proceedings of the Evaluation and Assessment on Software Engineering, EASE '19, pp. 340–345. ACM, New York, NY, USA (2019). DOI 10.1145/3319008.3319354. URL <http://doi.acm.org/10.1145/3319008.3319354>
3. Balachandran, V.: Reducing human effort and improving quality in peer code reviews using automatic static analysis and reviewer recommendation. In: Proc. of ICSE, pp. 931–940. IEEE Press (2013)
4. Chen, L., Wu, W., He, L.: How personality influences users' needs for recommendation diversity? In: CHI '13 Extended Abstracts on Human Factors in Computing Systems, CHI EA '13, pp. 829–834. ACM, New York, NY, USA (2013). DOI 10.1145/2468356.2468505. URL <http://doi.acm.org/10.1145/2468356.2468505>
5. Facebook: mention-bot (2015). Accessed: 2019-05-23. <https://github.com/facebookarchive/mention-bot>
6. Fejzer, M., Przymus, P., Stencel, K.: Profile based recommendation of code reviewers. Journal of Intelligent Information Systems 50(3), 597–619 (2018). DOI 10.1007/s10844-017-0484-1. URL <https://doi.org/10.1007/s10844-017-0484-1>
7. Ferwerda, B., Yang, E., Schedl, M., Tkalcic, M.: Personality traits predict music taxonomy preferences. In: Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems, CHI EA '15, pp. 2241–2246. ACM, New York, NY, USA (2015). DOI 10.1145/2702613.2732754. URL <http://doi.acm.org/10.1145/2702613.2732754>
8. Github: Understanding the github flow (2013). Accessed: 2019-05-23. <https://guides.github.com/introduction/flow/>
9. Github: Github api v3 (2016). Accessed: 2019-05-23. <https://developer.github.com/v3/>
10. GitHub: About code owners (2017). Accessed: 2019-05-23. <https://help.github.com/articles/about-codeowners/>
11. Github: request review in github (2017). Accessed: 2019-05-23. <https://help.github.com/articles/requesting-a-pull-request-review/>
12. Gousios, G., Pinzger, M., Deursen, A.v.: An exploratory study of the pull-based software development model. In: Proc. of ICSE, pp. 345–355. ACM (2014)
13. Gousios, G., Storey, M.A., Bacchelli, A.: Work practices and challenges in pull-based development: The contributor's perspective. In: 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE), pp. 285–296 (2016). DOI 10.1145/2884781.2884826
14. Gousios, G., Storey, M.A., Bacchelli, A.: Work practices and challenges in pull-based development: the contributor's perspective. In: Proc. of ICSE, pp. 285–296. ACM (2016)
15. Gousios, G., Zaidman, A., Storey, M.A., Van Deursen, A.: Work practices and challenges in pull-based development: the integrator's perspective. In: Proc. of ICSE, pp. 358–368. IEEE Press (2015)
16. Hoaglin, D.C., Iglewicz, B., Tukey, J.W.: Performance of some resistant rules for outlier labeling. Journal of the American Statistical Association 81(396), 991–999 (1986)

17. Jiang, J., He, J.H., Chen, X.Y.: Coredevrec: Automatic core member recommendation for contribution evaluation. *Journal of Computer Science and Technology* **30**(5), 998–1016 (2015)
18. Lee, M.K., Kusbit, D., Metsky, E., Dabbish, L.: Working with machines: The impact of algorithmic and data-driven management on human workers. In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, CHI '15*, pp. 1603–1612. ACM, New York, NY, USA (2015). DOI 10.1145/2702123.2702548. URL <http://doi.acm.org/10.1145/2702123.2702548>
19. Pham, R., Singer, L., Liskin, O., Filho, F.F., Schneider, K.: Creating a shared understanding of testing culture on a social coding site. In: *2013 35th International Conference on Software Engineering (ICSE)*, pp. 112–121 (2013). DOI 10.1109/ICSE.2013.6606557
20. Rahman, M.M., Roy, C.K., Collins, J.A.: Correct: Code reviewer recommendation in github based on cross-project and technology experience. In: *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*, pp. 222–231 (2016)
21. erlend sh: Only activate mention bot on prs without comments (2016). Accessed: 2016-09-16. <https://github.com/facebook/mention-bot/issues/119>
22. Sinha, R., Swearingen, K.: The role of transparency in recommender systems. In: *CHI '02 Extended Abstracts on Human Factors in Computing Systems, CHI EA '02*, pp. 830–831. ACM, New York, NY, USA (2002). DOI 10.1145/506443.506619. URL <http://doi.acm.org/10.1145/506443.506619>
23. Stolze, M., Nart, F.: Well-integrated needs-oriented recommender components regarded as helpful. In: *CHI '04 Extended Abstracts on Human Factors in Computing Systems, CHI EA '04*, pp. 1571–1571. ACM, New York, NY, USA (2004). DOI 10.1145/985921.986147. URL <http://doi.acm.org/10.1145/985921.986147>
24. Thongtanunam, P., Tantithamthavorn, C., Kula, R.G., Yoshida, N., Iida, H., Matsumoto, K.i.: Who should review my code? a file location-based code-reviewer recommendation approach for modern code review. In: *Proc. of SANER*, pp. 141–150. IEEE (2015)
25. Tsay, J., Dabbish, L., Herbsleb, J.: Influence of social and technical factors for evaluating contribution in github. In: *Proc. of ICSE*, pp. 356–366. ACM (2014)
26. Webhooks: webhooks (2017). Accessed: 2017-09-01. <https://developer.github.com/webhooks/>
27. Xia, X., Lo, D., Wang, X., Yang, X.: Who should review this change?: putting text and file location analyses together for more accurate recommendations. In: *Proc. of ICSME*, pp. 261–270. IEEE (2015)
28. Yu, Y., Wang, H., Filkov, V., Devanbu, P., Vasilescu, B.: Wait for it: Determinants of pull request evaluation latency on github. In: *Proc. of MSR*, pp. 367–371. IEEE (2015)
29. Yu, Y., Wang, H., Yin, G., Wang, T.: Reviewer recommendation for pull-requests in github: What can we learn from code review and bug assignment? *Information and Software Technology* **74**, 204–218 (2016)
30. Zanjani, M., Kagdi, H., Bird, C.: Automatically recommending peer reviewers in modern code review. *Transactions on Software Engineering* **42**, 530–542 (2016)