Exploring How Software Developers Work with Mention Bot in GitHub

Zhenhui Peng
The Hong Kong University of Science and Technology
HKSAR, China
zpengab@connect.ust.hk

Sunghun Kim
The Hong Kong University of Science and Technology
HKSAR, China
hunkim@cse.ust.hk

Jeehoon Yoo
Data Team, RIDI Corporation
Seoul, NA, Republic of Korea
jeehoon.yoo@ridi.com

Xiaojuan Ma
The Hong Kong University of Science and Technology
HKSAR, China
mxj@cse.ust.hk

Meng Xia
The Hong Kong University of Science and Technology
HKSAR, China
iris.xia@connect.ust.hk

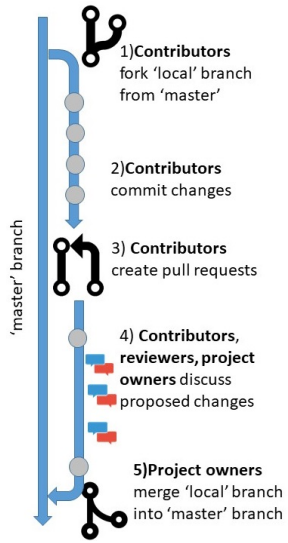

Figure 1: The pull process

ChineseCHI '18, April 21–22, 2018, Montreal, QC, Canada
ACM 978-1-4503-6508-6/18/04.
https://doi.org/10.1145/3202667.3202694

Abstract
Recently, major software development platforms have started to provide automatic reviewer recommendation (ARR) services for pull requests, to improve the 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 the bot's 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 effort, but are bothered by its unstable setting and unbalanced workload allocation. We conclude with design considerations for improving ARR services.

Author Keywords
Automatic reviewer recommendation services; mixed-methods; user experience; software development platform.

ACM Classification Keywords
H.5.m [Information interfaces and presentation (e.g., HCI)]: Miscellaneous

	S1	S2
Archival data (# of projects)	205	227
Issue comments (# of comments)	53	90
Surveys (# of respondents)	52*	34*
Interviews (# of interviewees)	-	6

Table 1: Data summary of two stages. *S1's respondents serve as both contributors and reviewers, while S2's respondents are either project owners, reviewers or contributors.

Response time	**#**
Reduced	25
Increased	6
No difference	124

Table 2: We compared the response time in the PR_{Bot} and $PR_{Non-bot}$ groups using Mann-Whitney-Wilcoxon test.

For archival analysis in Stage I, we exclude projects with less than four contributors (since the mention-bot normally recommends up to three candidates) and irrelevant PRs (e.g. not closed). We finally have 64,937 PRs (9,413 call mention-bot) from 155 projects.

Introduction

More and more developers work collectively on software development projects in online platforms such as GitHub. GitHub uses the pull-based development model to implement the peer review for collaborative coding, which normally involves three kinds of developers: contributors, reviewers and owners (Figure 1). When contributors push their changes of code to the project repository, pull requests (PRs) are issued and they must be inspected by reviewers and approved by owners before the new code gets merged. Hence, there is an increasing demand for timely and qualified PR reviews [3]. Automatic reviewer recommendation(ARR) services, such as Facebook mention-bot which generates its recommendations using code change history, were introduced to streamline the process.

Existing research on ARR services mainly focus on improving their accuracy, response time and response rate [1, 3]. However, little research has looked into the user experiences of and interaction among different stakeholders involved in the process (Figure 1), which has a great influence on designing usable recommendation services. For example, Marlow et al. [2] found that GitHub reviewers tend to examine contributors' profiles before deciding on whether to accept or reject the PR review request. Despite the insightful findings, they did not study the needs from the contributors' perspective and the process did not involve ARR. In this paper, we use the mention-bot in GitHub as a lens to explore *1) how developers perceive and work with mention-bot in practice*, and *2) what are the most critical needs for different types of users involved in ARR services.*

Two-Stage Mixed-Method Investigation

We take a two-stage mixed-method approach to better understand user experience (Table 1). In Stage I (2015.11 to 2016.06), we identify 205 projects employing mention-bot. In Stage II (2016.07 to 2017.08), we revisit the 205 projects as well as 22 projects that newly employ mention-bot (Table 4). In both stages, to gain an in-depth understanding of why people use/do not use mention-bot and what they expect from an ARR service, we collect related issue comments in GitHub and invite GitHub developers by email to take our surveys or interviews.

Stage I. (1)Performance. We first investigate the response rate and response time ($T_{1stResponse} - T_{SubmitPR}$) of reviewers suggested by mention-bot based on archival data. For each PR that mention-bot comments on, we count it as a hit if at least one of the recommended reviewers show up. Overall the average response rate across the 155 projects is 75.37% (SD = 26.92%). For each project, we divide PRs into the two groups by whether the mention-bot is called, and compare their average response time (Table 2). The average response time of the six projects whose $PRTime_{Non-bot} < PRTime_{Bot}$ is about 1.7 hours (SD = 5.14), while that of the 25 projects whose $PRTime_{Non-bot} > PRTime_{Bot}$ is around 9.45 hours (SD = 74.01). This suggests that mention-bot is more likely to reduce the response time in less active projects. The survey mainly asks participants about mention-bot's perceived usefulness and their favorite features(Figure 2). Respondents agree that mention-bot recommends appropriate reviewers (about 75.0%), and the reduction in effort in identifying reviewers is appreciated most (71.1%). **(2)Likability**. Developers show diverse likability of the mention-bot in the issue comments (25 positive, 20 negative and 8 neutral). The developers seem to like its core features "Reviewer recommendation" and "Automatic notification" (favored by 83.1% and 60.4% in the surveys, respectively). However, some developers *"don't want to receive any notifications for this repository as they are not collaborators there"*(C1). Apart from its insensitivity to context, mention-bot's unbalanced workload allo-

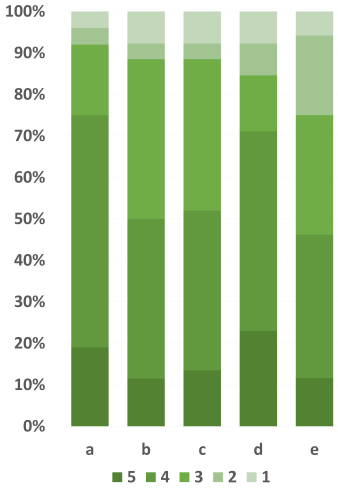

Figure 2: Perceived usefulness in Stage I. **a.**Appropriateness **b.**Contributors get faster response **c.**Reviewers respond faster **d.**Save efforts to identify reviewers & **e.** Explore PRs. Point from 5 to 1 represent from *Strongly agree* to *Strongly disagree*.

Status	# of projects
Removed	22
Still use	30
Disappeared	11
No records	142*
Newly add	22

Table 4: Mention-bot's status in the projects in Stage II. * "No records" means that no words in the project explicitly indicate that it is still running or it has been removed.

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

Table 3: Contents showed in some comments in Stage II

cation is a burden to some reviewers and discourages others (*"It's almost always recommending the same person in our project which is not really that helpful."*(C2)). The "Enable/disable notification for certain PRs/people" feature of the mention-bot is designed to resolve the aforementioned issues, but its unfriendly configuration design intimidates many users (only 37.5% of the survey respondents like it).

Stage II. Perception of different user groups. Table 3 summarizes some of the 90 related issue comments in Stage II. Across all user groups in the survey, we ask respondents to rate the perceived usefulness and annoyance of the mention-bot as well as the efficacy of each feature on a 5-point Likert scale (1 being the least). Overall, mention-bot users "find it useful" (4.08, SD = 0.64). The project owners employ mention-bot mostly for its "efficiency" (4.29, SD = 0.95) . With mention-bot, owners spend less effort in "managing the PR process" (3.71, SD = 0.76) and can "engage developers more in the projects" (3.86, SD = 0.69). Also, 70% of non-user respondents hope that the projects they participate in would employ it. Reviewers who use it agree that "it saves effort in finding proper PRs". However, contributors who use mention-bot do not always think that they can "get faster responses from its recommended reviewers than from others" (3.00, SD = 0.71), or that "the suggested reviewers certainly provide better feedback" (3.20, SD = 1.10), or that "it improves interaction with other developers" (3.20, SD = 1.10). However, they do agree that it saves efforts in looking for proper reviewers (4, SD = 1.22), which is consistent with our findings in Stage I. Six of the survey respondents (I1,2,3,4,5,6) join our semi-structural online interview and share their user experience with or without mention-bot (Table 5). Surprisingly, the reason I3,4,5,6 removed mention-bot are quite similar (*"GitHub added the "suggested reviewers" feature which is enough for our needs"*(I3)). The "suggested reviewers" feature is plugged into the GitHub platform so that users do not need to configure it by themselves and worry about its instability (which bothers mention-bot users as showed in Table 3). However, interviewees also commented that "suggested reviewers" is not flexible enough for big projects (*"The feature needs permission. The suggested reviewers should be members of our project, but we want all the 2000 contributors in our project, and mention-bot suits our needs"*(I1)).

Critical Factors and Design Consideration

Simplicity and Stability for Project Owners. We find that most of the projects just keep the default setting of mention-bot, and sometimes owners cannot configure it well (*"The configuration added is not working, so I just removed it..."*(C3)). Since they tend to *"only care about the PRs and want the bot to simply tell what its capacities are"*

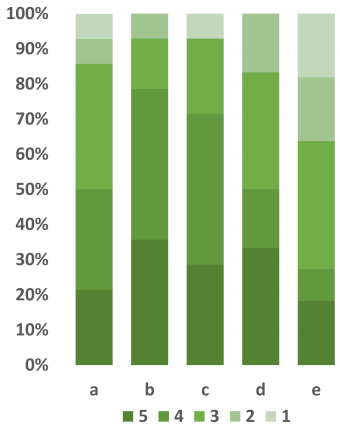

Figure 3: Perceived usefulness of features in Stage II. **a.**Message Customization **b.**Explanation of result **c.**List of recommended reviewers **d.**Delayed time **e.** Blacklist. Point from 5 to 1 represent from *Extremely useful* to *Not useful at all*.

I1	project owner, user
I2	reviewer&contributor, non-user
I3,4, 5,6	project owners, used but removed it later

Table 5: Status of interviewees. Note that people accepting our request are mostly project owners, which indicates that they are more concerned about the experience of ARR services.

(I1), we propose that a better ARR service should have an easier configuration process. For example, the service can have a log so that project owners can save their settings and easily reset the configuration to a suitable and stable state. If the project needs more external contributions, the owner can use a shortcut to adjust some options to invite external reviewers.

Better Transparency for Contributors. According to our survey in Stage II, contributors tend to check on mention-bot's recommendation, and they call for information that can improve their understanding of why a particular recommendation is made. *List of recommended reviewers*(78.6%) and *Explanation of the result*(71.5%) are perceived as useful (Figure 3). Therefore, we propose that a better ARR service should keep their recommendations transparent to contributors, especially regarding the *qualification* and *availability* of the suggested reviewers (I2). In case contributors would like to manually select reviewers, the ARR service can provide a ranked list of the potential reviewers, each with a brief profile summarizing their role in the project, specialty, recent activeness, current workload, and so on. **Flexible Notification Preference Settings for Reviewers.** Reviewers are bothered the most by excessive PR review notifications. When reviewers are already overloaded, they *"do not want to receive more review requests"* (I2). However, only project owners can set the rules to filter reviewers in the candidate pool in mention-bot. Hence, we propose that a better ARR service should allow reviewers to specify personal notification preferences on their side. Reviewers can change their status to "Do not disturb" when occupied, declare types of uninterested PRs, and set a maximum PR quota. In addition, similar to suggestions showed in the comments (Table 3), the notifications can provide useful links and a brief introduction of the contributors.

Conclusion and Future Work

In this paper, we explore how people work with mention bot in GitHub to give hints on evaluating user experience of ARR services. We use a two-stage mixed-method approach to investigate the practical usefulness of mention bot and the critical needs of different types of users. Our Stage I investigation showed that mention bot saved contributors' effort in identifying proper reviewers (75.0%). In Stage II, we found that mention bot's adoption suffers from bugs, unbalanced workload allocation and other ARR alternatives. According to our survey and interview with project owners, contributors and reviewers, we propose a set of considerations for designing more user-friendly ARR services. In the future, we plan to improve the coverage and generality of our study, develop an ARR service based on the findings, and test its user experience in the wild.

REFERENCES

1. Jing Jiang, Jia-Huan He, and Xue-Yuan Chen. 2015. CoreDevRec: Automatic Core Member Recommendation for Contribution Evaluation. *Journal of Computer Science and Technology* 30, 5 (2015), 998–1016.

2. Jennifer Marlow, Laura Dabbish, and Jim Herbsleb. 2013. Impression Formation in Online Peer Production: Activity Traces and Personal Profiles in Github. In *Proc. CSCW '13*. ACM, New York, NY, USA, 117–128. DOI: http://dx.doi.org/10.1145/2441776.2441792

3. Yue Yu, Huaimin Wang, Gang Yin, and Tao Wang. 2016. Reviewer recommendation for pull-requests in GitHub: What can we learn from code review and bug assignment? *Information and Software Technology* 74 (2016), 204–218.