Exploring the Use of @-mention to Assist Software Development in GitHub

Yang Zhang, Huaimin Wang, Gang Yin, Tao Wang, Yue Yu
Key Lab. of Parallel and Distributed Computing, College of Computer, National University of Defense Technology, Changsha, 410073, China
{yangzhang15, hmwang, jack.nudt, taowang2005, yuyue}@nudt.edu.cn

ABSTRACT

Recently, many researches propose that social media tools can promote the collaboration among developers, which are beneficial to the software development. Nevertheless, there is little empirical evidence to confirm that using @-mention has indeed a beneficial impact on the issues in GitHub. In this paper, we analyze the data from GitHub and give some insights on how @-mention is used in the issues (*general-issues* and *pull-requests*). Our statistical results indicate that, @-mention attracts more participants and tends to be used in the difficult issues. @-mention favors the solving process of issues by enlarging the visibility of issues and facilitating the developers' collaboration. In addition to this global study, our study also build a *@-network* based on the @-mention database we extract. Through the *@-network*, we can mine the relationships and characteristics of developers in GitHub's issues.

CCS Concepts

•**Human-centered computing** → *Collaborative and social computing;* •**Software and its engineering** → Empirical software validation;

Keywords

Issues; Social media; @-mention; GitHub

1. INTRODUCTION

GitHub is a social collaborative software development community. The platform integrates many social media tools involving follow [1], watch [1], comment action [2] and @-mention [3]. @-mention is a typical social media used in the online social platform. It allows users to reference a specific user by simply placing an "@" symbol in front of the username they wish to reference [4]. Compared to follow, watch and other general social media like wikis [5], blogs [6] and microblogs [7], @-mention usually comes from the issue's description body or comments in GitHub, which makes it more deeply involved in the solving process of issues.

Previous work has identified that social media tools are widely used in the software development context. These social media tools make it possible to leverage articulated social networks and observed code-related activity simultaneously, which supports the type of awareness that only available to core developers in previous [2]. Social media has changed the way that people collaborate and share information [8]. Basically, these researches mainly focused on the correlation between the general social media and the overall software development. Other researches proved that @-mention is a strong predictor of information diffusion [9] and is a significant factor in enlarging the visibility of a post and helping initiate responses and conversations [10]. Nevertheless, little effort has been done on analyzing how @-mention is used in a population of issues and on whether their use has any impact on the solving process of issues in GitHub.

In this paper, we use qualitative and quantitative approaches to conduct an exploratory study of @-mention usage in GitHub. Our results give an explicit description of the current usage of @-mention and elicit some important implications for the developers to know better of @-mention used in the issues of GitHub. In particular, we have studied (1) how @-mention is used in GitHub's issues (*general-issues* and *pull-requests*) and (2) how @-mention may influence the solving process of issues (the number of comments, the time to solve, *etc.*). Our results show that there exists a significant difference in terms of the characteristic of issues between issues that do not have @-mention and that do. @-mention tends to be used in the difficult issues and it can reduce the time-to-solve as well as the time delay between comments in case of the same difficulty of issues. Our investigation proves that @-mention has a positive impact on the solving process of issues by enlarging the visibility of issues and facilitating the developers' collaboration. Besides this global study, we also build a *@-network* and conduct a more fine-grained preliminary study which aims at studying how use @-mention to mine the relationships and characteristics of developers in GitHub's issues.

The remainder of this paper is structured as follows. Section 2 describes the related concepts and our research questions. In Section 3, we introduce our analysis and the main results. Section 4 discusses the results and presents further analysis steps on the topic. Section 5 reports on the related work and threats to validity are discussed in Section 6. We conclude the article in Section 7.

Internetware '15, November 06 2015, Wuhan, China

DOI: http://dx.doi.org/10.1145/2875913.2875914

2. PRELIMINARIES & PROBLEM DEFINITION

In this section, we give a brief introduction of issues and @-mention in GitHub. Then we propose our research questions.

2.1 Issues

Open-Source Software (OSS) enables anyone to be part of the development process as large [11]. Reporting issues (bugs, new features or requirements) may be the most common one to contribute among the different ways (*e.g.* testing, coding, *etc.*) [12]. These issues are managed by the issue trackers, which aim at facilitating the management of issues, by providing a feature-rich interface.

GitHub is the largest social collaborative software development community. It is a developer-friendly environment integrating many functionalities, including wiki, issue tracker, and code review [13]. GitHub provides a light-weight and flexible issue tracker, which provides the usual facilities in issue tracking, such as filing issue tickets, labeling them, setting the milestone and submitting the comments. In our study, we divide the issues into two kinds, *general-issues* and *pull-requests*. As the examples shown in Figure 1, issue *#16831* is a *general-issue* and issue *#18936* is a *pull-request*. There are some differences between the interfaces of *general-issues* and *pull-requests*. *Pull-requests* as implemented by GitHub in particular, is a new model for collaborating on distributed software development [14]. It is also maintained by the issue tracker in GitHub. For each opened *pull-request*, an issue is opened automatically. So every *pull-request* is an issue, but not every issue is a *pull-request*. We can consider the *pull-requests* as special issues in addition to the *general-issues*. Different from the *general-issues*, *pull-requests* have other two types of comments beside *general comments* (basic type of comments on *general-issues* or *pull-requests*): (1) *pull-request review comments* (comments on the portion of diff patch in *pull-requests*) and (2) *commit comments* (comments on the commits of *pull-requests*).

2.2 @-mention

@[1], normally read as "at", especially in email addresses, is the meaning of "located at" or "directed at". In recent year, more and more online social platform (Facebook[2], Twitter[3], *etc.*) use "@" to denote a reference or a reply which we called as @-mention. The feature of @-mention enables users to directly reference others by putting an "@" symbol before their username such as "@Jack". Then @-mention can automatically interpret these as links to the user's profile. In addition to the link function, after @-mentioning somebody, the @-mentioned person could receive a reminder to help himself respond immediately. In the issues of GitHub, @-mention can be used in the description body or the comments of issues. As the "@" symbol exists in issues' title is just a text and does not have the link function, we do not consider it in our study. Figure 2 shows an example about how @-mention be used in GitHub. In issue *#21290*, when *zeckalpha* reports this issue, he @ *robertomiranda* for review in his issue's description body. After *robertomiranda* reviewed this new issue, he then @ *dhh* and *guilleiguaran* for advice in this issue's comment.

[1]http://en.wikipedia.org/wiki/@
[2]https://www.facebook.com/
[3]https://www.twitter.com/

(a) *general-issues*

(b) *pull-requests*

Figure 1: Two examples of issues in GitHub

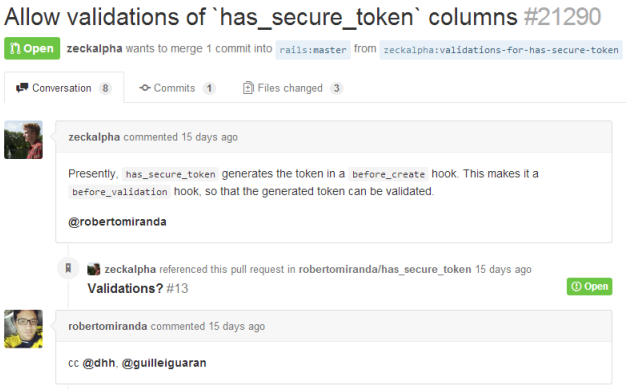


Figure 2: An example of @-mention used in GitHub

2.3 Research Questions

In our investigation of @-mention, we focus on the following research questions:

RQ1: @-mention usage. To what extent is @-mention used in the issues of GitHub?

For answering this question, we choose a famous and large project for our investigation. Since issues consist of *general-issues* and *pull-requests* as described before, we respectively analyze the usage of @-mention in *general-issues* and *pull-requests*, including the number of use in each issue, usage locations, usage scenarios as well as the @-mentioned developers information *etc.*

RQ2: @-mention influence. What kind of differences are there between the issues with and without @-mention? Does using @-mention influence the solving process of issues?

In answer to this question, we mainly focus on comparing differences between issues with @-mention and without @-

mention on the basis of the following characteristics of issues: the number of comments, the number of participants, and the time to solve *etc.* Then we use the statistics tests to verify the significance of these differences.

3. USE OF @-MENTION IN RAILS

We perform our exploratory study on a famous project called Ruby on Rails[4] (Rails), which is one of the largest and most successful projects hosted on GitHub. Rails is also one of the great open source projects that GitHub is using to power its infrastructure. Rails is a web-application framework that includes everything needed to create database-backed web applications, according to the Model-View-Controller (MVC) pattern. Table 1 shows the basic information of Rails.

Table 1: The basic information of Rails

language	created_at	watch	star	fork
Ruby	2008-04-11	2010	27534	11055

Our research database relies on the GitHub Rest API[5], which we can use to retrieve information on publicly accessible contents of the Rails repository. Up to August 2015, we collect 21273 issues (*general-issues*: 7502, *pull-requests*: 13771), involving 122935 comments (*general comments*: 93711, *commit comments*: 12495, *pull-request review comments*: 16729) and 10094 developers.

In order to analyze the @-mention, we need to extract the information of @-mention (*e.g.* the @-mentioned developer, the @-mention location *etc.*) from the textual data of issues (description body and comments). As shown in Algorithm 1, the extraction work can be divided into 4 steps: (1) First step, for each issue in Rails, we extract its description body and *general comments* information to build the basic textual data. We define these textual data as the *issueText*. (2) Second step, if this issue belongs to the *pull-requests*, we extract its *pull-request review comments* and *commit comments* and add them into the *issueText*, otherwise we directly go to the third step. (3) Third step, we judge whether the *issueText* contains at least one "@" symbol. If the *issueText* contains "@", we use the regular expression method to extract the @-mentioned developer information, *i.e.* the username in the string "@username", otherwise we scan the next issue. (4) Fourth step, we query the *Project Users* table to check whether the "@" is a valid @-mention operation. Because some text in back of "@" are not real username, such as "Hongli Lai <hongli@phusion.nl>", which is a email address. If it is a valid @-mention operation, then we insert the valid @-mention information into our MySQL database for the manipulation in the subsequent phases, otherwise we scan the next issue.

Algorithm 1 Extracting @-mention method

Input: *Issues* //*general-issues* or *pull-requests*

@-mention database ← ∅
foreach *issue* I_a **in** *Issues* **do**:
  descriptionText ← extractDescription(I_a)
  generalCommentsText ← extractGeneralComments(I_a)
  issueText ← *descriptionText* ∩ *generalCommentsText*

  if $I_a \in$ *pull-requests*:
    pullRequestReviewCommentsText ← extractPullRequestReviewComments(I_a)
    commitCommentsText ← extractCommitComments(I_a)
    issueText ← *issueText* ∩ *pullRequestReviewCommentsText* ∩ *commitCommentsText*

  if @ ∈ *issueText*:
    atDeveloperData ← extractAtDeveloperData(*issueText*)
    foreach *developer* D_a **in** *atDeveloperData* **do**:
      if $D_a \in$ *Project Users*:
        @-mention database ← $\{I_a, D_a\}$

Output: *@-mention database*

3.1 @-mention Usage

As a first step, we study the ratio of issues with at least one @-mention operation. After parsing the *issueText* by the Algorithm 1, we extract 41395 @-mention operations in Rails project. 13784 (33%) of them are extracted from the *general-issues* and 27611 (67%) are extracted from the *pull-requests*. We find that, from 21273 studied issues, there are 11124 (52%) of them containing @-mention. Specifically, there are 3833 (51%) of the total 7502 *general-issues* and 7291 (53%) of the total 13771 *pull-requests* containing @-mention. This result reveals that **most @-mention are used in the *pull-requests*, and no matter in *general-issues* or in *pull-requests*, @-mention are used quite a lot (nearly half)**.

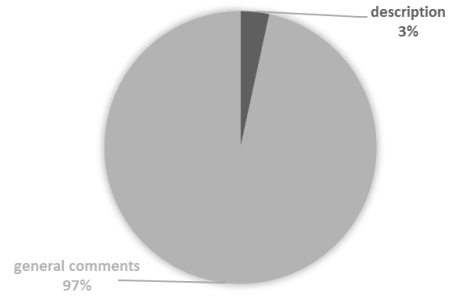
We then study how many @-mention on average are used in each issue (*general-issue* or *pull-request*) that has @-mention operation. Table 2 shows the main statistical results. The first column (#@-mention) shows the number of @-mention operation used in each issue. According to the second column (#general-issue), third column (#pull-request) and fourth column (#issue), we can find that the vast majority of issues only have 1 to 3 @-mention operations (nearly 65%). This result reveals that **the frequency of each issue using @-mention is low**.

Table 2: The number of @-mention usage in issues

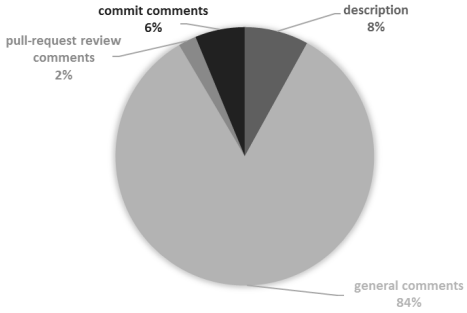
#@-mention	#general-issue	#pull-request	#issue
1	1232 (32.14%)	2380 (32.64%)	3612 (32.47%)
2	796 (20.77%)	1449 (19.87%)	2245 (20.18%)
3	491 (12.81%)	908 (12.45%)	1399 (12.58%)
4	364 (9.50%)	652 (8.94%)	1016 (9.13%)
5	226 (5.90%)	435 (5.97%)	661 (5.94%)
6	157 (4.10%)	318 (4.36%)	475 (4.27%)
7	143 (3.73%)	254 (3.48%)	397 (3.57%)
8	93 (2.43%)	199 (2.73%)	292 (2.62%)
9	82 (2.14%)	140 (1.92%)	222 (2.00%)
10	64 (1.67%)	97 (1.33%)	161 (1.45%)
>10	185 (4.83%)	459 (6.30%)	644 (5.79%)
Total	3833	7291	11124

As mentioned in Section 2, @-mention is usually used in the issue's description body or *general comments*. When in *pull-requests*, @-mention is also used in *pull-request review comments* or *commit comments*. So we study the specific location of @-mention used in issues. Figure 3-a shows that, in *general-issues*, 97% @-mention are used in the *general comments* and only 3% @-mention are used in the description body. Figure 3-b shows that 92% @-mention are used in the comments (*general comments*: 84%, *commit comments*: 6%, *pull-request review comments*: 2%) and only 8% @-mention are used in the description body in the *pull-requests*. This result indicates that **most @-mention are used in issue's comments rather than issue's description body, *i.e.* @-mention is more likely to be used in the conversation of developers**.

[4]https://github.com/rails/rails
[5]http://developer.github.com/

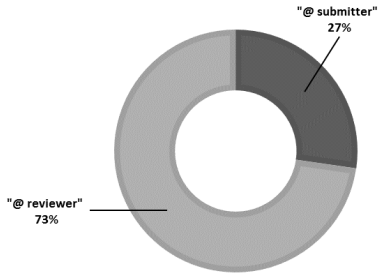


(a) *general-issues*

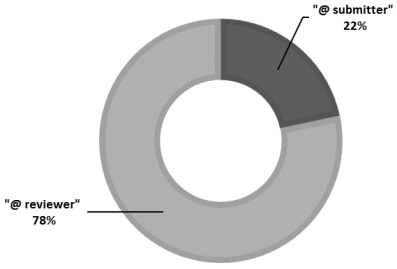


(b) *pull-requests*

Figure 3: The distribution of specific location of @-mention used in issues



(a) *general-issues*



(b) *pull-requests*

Figure 4: The distribution of specific scenario of @-mention used in issues

Further more, we divide the scenarios of @-mention used in issues into two kinds, "@ submitter" and "@ reviewer". "@ submitter" means the reviewer participated in the issue's discussion @ the issue's submitter and "@ reviewer" means the issue's submitter @ the reviewer. We find that 24% @-mention are used for "@ submitter" while 76% @-mention are used for "@ reviewer" in issues. The specific distribution in *general-issues* and *pull-requests* are shown in Figure 4. This result indicates that **most @-mention are used by the issue's submitters to @ other developers for review**.

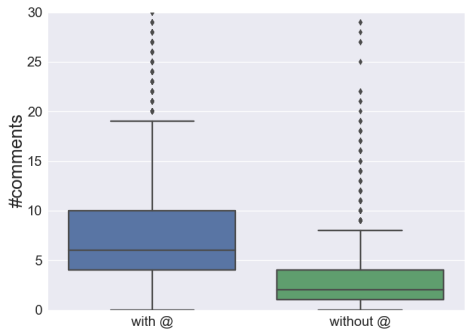
As GitHub is a social coding site, among the total 10094 developers in Rails, 10057 of them are not internal project members but also contribute their codes and suggestions to the only 37 internal project members. The internal project members can access to the repository of Rails and they manage all the issues from internal or external developers. In our study, we find that 54% @-mentioned developers are internal project members. So each internal project member on average can be @-mentioned 606 times, while the average time of each external developer be @-mentioned is only 2 times. Specifically, 42% @-mentioned developers are internal project members in *general-issues*, while the ratio is raised to 60% in *pull-requests*. This result proves that **@-mention is generally used to @ the internal project members and this target-oriented phenomenon in *pull-requests* is more obvious than in *general-issues***.

3.2 @-mention Influence

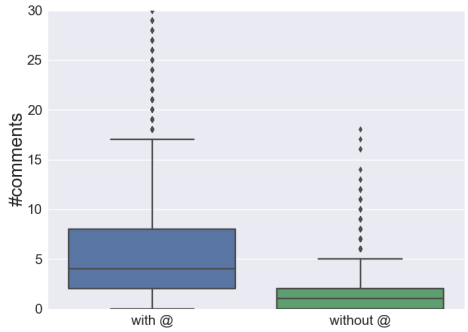
Based on the investigation of the usage of @-mention in issues, in this section, we try to analyze if @-mention has a positive impact on the solving process of issues. We use the R statistical analysis tool to find the differences between issues with @-mention and without @-mention. We also use statistical tests including Mann-Whiney-Wilcoxon (MWW) test, Z test and Cliff's δ to validate the significance of these differences. All of these statistical tests are non-parametric statistical hypothesis tests. They do not assume any specific distribution, which is a suitable property for our experimental analysis.

First, we study the distribution of the number of comments in issues with @-mention and without @-mention. As shown in Figure 5-a, the average number of comments is 3.0 (median: 2.0) for *general-issues* without @-mention, while the value is raised to 8.7 (median: 6.0) for *general-issues* with @-mention. Figure 5-b shows that, the average number of comments is 1.1 (median: 1.0) for *pull-requests* without @-mention, while the value is raised to 6.2 (median: 4.0) for *pull-requests* with @-mention. Using the statistical tests, we verify that the differences between issues with @-mention and without @-mention are statistically significant (*general-issues*: p<2.2e-16, z=32.7 [very significant], δ=0.67 [large]; *pull-requests*: p<2.2e-16, z=59.3 [very significant], δ=0.72 [large]). This result indicates that **issues with @-mention are likely to have more comments than issues without @-mention, *i.e.* @-mention promotes the discussion of developers in issues**.

Then, we study the distribution of the number of participants in issues with @-mention and without @-mention. As shown in Figure 6-a, the average number of participants is 2.2 (median: 2.0) for *general-issues* without @-mention, while the value is raised to 4.5 (median: 4.0) for *general-issues* with @-mention. Figure 6-b shows that, the average number of participants is 1.7 (median: 2.0) for *pull-requests* without @-mention, while the value is raised to 3.7 (median:

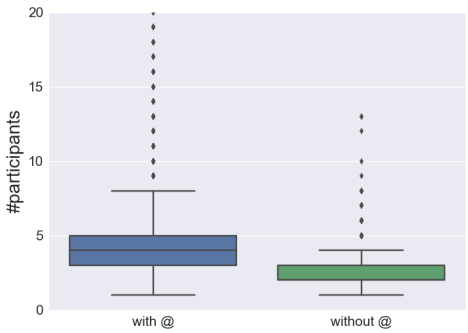


(a) *general-issues*

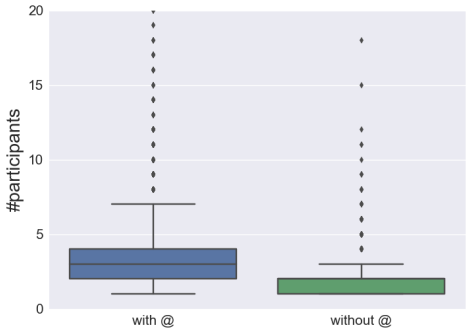


(b) *pull-requests*

Figure 5: The number of comments in issues with @-mention and without @-mention[6]



(a) *general-issues*



(b) *pull-requests*

Figure 6: The number of participants in issues with @-mention and without @-mention

3.0) for *pull-requests* with @-mention. We test and confirm that the distributions between issues with @-mention and without @-mention are significantly different using the statistical tests (*general-issues*: p<2.2e-16, z=45.4 [very significant], δ=0.67 [large]; *pull-requests*: p<2.2e-16, z=56.8 [very significant], δ=0.67 [large]). This result indicates that **issues with @-mention are likely to have more participants than issues without @-mention, *i.e.* @-mention facilitates developers to participate in the solving process of issues**.

And then we study the distribution of the time-to-solve (time interval between a issue is opened and closed) in issues with @-mention and without @-mention. In our study, we only focus on the 20371 (95.8% of total 21273 issues) closed issues (*general-issues*: 7316, *pull-requests*: 13235). As shown in Figure 7-a, the average time to solve the *general-issues* with @-mention is 2167.6 hours (median: 166.0 hours), while the time is dropped to 541.5 hours (median: 11.0 hours) for *general-issues* without @-mention. As shown in Figure 7-b, the average time to solve the *pull-requests* with @-mention is 894.3 hours (median: 21.0 hours), while the time is dropped to 183.0 hours (median: 1.0 hours) for *pull-requests* without @-mention. In our statistical tests, the results prove that these differences are statistically significant (*general-issues*: p<2.2e-16, z=23.0 [very significant], δ=0.39 [medium]; *pull-requests*: p<2.2e-16, z=22.4 [very significant], δ=0.44 [medium]). This result indicates that **issues with @-mention are likely to need more time to deal with than issues without @-mention**.

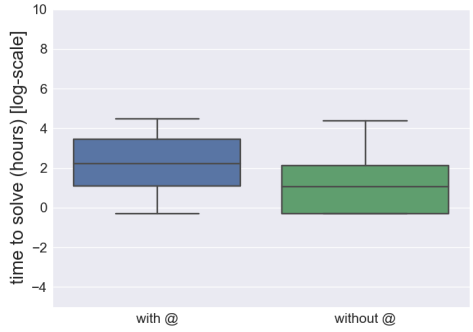
We think the number of participants is an important factor to illustrate the difficulty of issues. Difficult issue may need longer time to solve than easy issue, *i.e.* difficult issue may need more participants to discuss than easy issue. In order to further investigate the impact of @-mention on issues, we compare the difference of time-to-solve and time-between-comments between issues with @-mention and without @-mention when they have the same number of participants. Time-between-comments reveals the time delay among the developers' discussion. We use the Spearman's rho (ρ) correlation coefficient to measure the strength of monotonic relationships between the considered attributes. The values of ρ range from -1 to 1, where a perfect correlation is represented either by a -1 or a 1, meaning that the variables are perfectly monotonically related. On the contrary, the closer to 0 the ρ is, the more independent the variables are.

Table 3 shows the main analysis results. The number of participants is 1 (only the issue's submitter), which means there are no comments in the issue, so the average time-to-

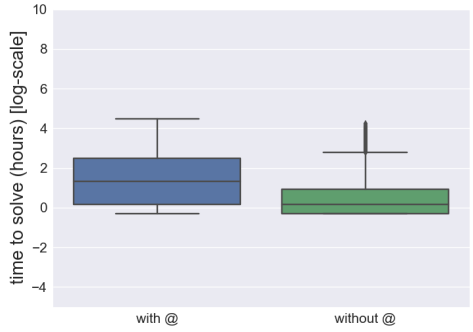
[6]In the boxplot, there are 5 main horizontal lines. From top to bottom, the top line indicates the max value. The second line indicates the upper quartile (25% of data points are above this line). The third line indicates the median value of the dataset. The fourth line indicates the low quartile (25% of data points are below this line). The bottom line indicates the min value. All data points above the top line or below the bottom line are outliers (determined by the tool).

Table 3: Difference between issues with @-mention and without @-mention in the case of different number of participants

#participants	general-issues						pull-requests					
	with @-mention			without @-mention			with @-mention			without @-mention		
	#issue	avg. time to solve (hours)	avg. time between comments (hours)	#issue	avg. time to solve (hours)	avg. time between comments (hours)	#issue	avg. time to solve (hours)	avg. time between comments (hours)	#issue	avg. time to solve (hours)	avg. time between comments (hours)
1	40 (5%)	369.28	369.28	814 (95%)	180.18	180.18	474 (13%)	71.26	71.26	3164 (87%)	49.71	49.71
2	605 (26%)	514.53	212.99	1755 (74%)	243.95	153.67	1831 (44%)	216.66	100.19	2313 (56%)	134.19	92.70
3	1009 (60%)	963.03	253.12	659 (40%)	887.47	314.71	1848 (76%)	496.00	137.83	584 (24%)	534.07	226.32
4	711 (78%)	1634.56	277.21	201 (22%)	1961.59	438.50	1163 (87%)	1083.18	197.52	173 (13%)	1128.95	288.31
5	470 (89%)	2156.31	387.03	61 (11%)	2899.69	403.66	648 (93%)	1592.13	226.08	47 (7%)	1845.34	319.82
6	241 (91%)	4122.99	501.29	25 (9%)	4194.44	574.36	363 (94%)	1951.63	267.94	22 (6%)	2171.68	357.41
>6	521 (96%)	5253.92	373.21	24 (4%)	5720.49	510.47	583 (96%)	2409.18	211.38	22 (4%)	3006.14	269.01
ρ		1.00	0.68		1.00	0.89		1.00	0.89		1.00	0.79



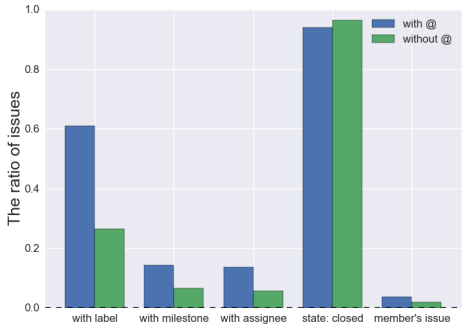
(a) *general-issues*



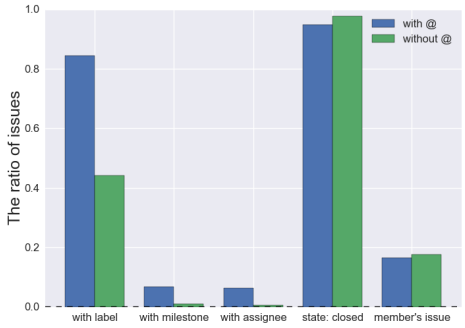
(b) *pull-requests*

Figure 7: The time-to-solve in issues with @-mention and without @-mention

solve is equal to the average time-between-comments. As can be seen, the ρ values reveal that on average the time-to-solve and the time-between-comments tend to increase together with the number of participants in *general-issues* and *pull-requests* and it may confirm that the number of participants has an impact on the issue's difficulty. The percentage of issues with @-mention indicates that **@-mention tends to be used in the difficult issues**. The time interval of *avg.* time-to-solve between issues with @-mention and without @-mentions reveals that **@-mention can shorten the time-to-solve in case of the same difficulty of issues (same number of participants)**. Similarly, the time interval of *avg.* time-between-comments between issues with @-mention and without @-mention reveals that **@-mention can cut down the delay among the comments, *i.e.* @-mention accelerates the conversation of participants**.



(a) *general-issues*



(b) *pull-requests*

Figure 8: Other characteristics of issues with @-mention and without @-mention

We also analyze other characteristics of issues with @-mention and issues without @-mention. As shown in Figure 8-a, we find that 60.9% *general-issues* with @-mention have label, while the ratio for *general-issues* without @-mention is only 26.3%. Similarly, 14.3% *general-issues* with @-mention have milestone and the ratio for *general-issues* without @-mention is 6.4%, 13.6% *general-issues* with @-mention have assignee and the ratio for *general-issues* without @-mention is 5.6%. For the state of issue (closed or open), 93.8% *general-issues* with @-mention are closed and the ratio for *general-issues* without @-mention is 96.5%. 3.6% *general-*

issues with @-mention are submitted by internal project members and the ratio for *general-issues* without @-mention is 1.8%. Figure 8-b shows the main results for *pull-requests*. We find that 84.4% *pull-requests* with @-mention have label, while the ratio for *pull-requests* without @-mention is only 44.1%. Similarly, 6.7% *pull-requests* with @-mention have milestone and the ratio for *pull-requests* without @-mention is 1.1%, 6.3% *pull-requests* with @-mention have assignee and the ratio for *pull-requests* without @-mention is 0.7%. For the state of issue, 94.8% *pull-requests* with @-mention are closed and the ratio for *pull-requests* without @-mention is 97.6%. 1.7% *pull-requests* with @-mention are submitted by internal project members and the ratio for *pull-requests* without @-mention is 1.8%. These results reveal that **issues with @-mention are more likely to have label, milestone as well as assignee than issues without @-mention**. As mentioned before, issues with @-mention need more time to solve than issues without @-mention, so the ratio of closed issues in issues with @-mention is little less than issues without @-mention.

4. DISCUSSION AND FURTHER WORK

Our study allows us to better characterize how @-mention is used in Rails. As the results indicate, @-mention is used quite a lot in Rails, especially in the *pull-requests*. Our explanation is that Rails is a famous and large open source project which attracts thousands of developers (internal or external) to contribute and discuss online. In particular, the mechanism of *pull-requests* makes the internal project members need much resource and time to decide whether the *pull-requests* should be merged into the core repository or not. Because the *pull-requests* need more discussion, @-mention is very likely to be used to involve more developers in the solving process. The specific location and scenario of @-mention reveal that, @-mention is more likely to be used in the comments during the developer's conversation instead of in the description body of issues and most @-mention are used for issue's submitter @-mentioning other reviewers. We consider it is difficult for the issue's submitter to @ suitable developers at the beginning of the issue's solving process. While during the conversation in the form of comments, with the assistance of other participants, the @-mention problem would be solved easily. Because GitHub provides a distributed collaborative software development pattern, all developers want their codes and suggestions to be accepted by the internal project members. And that is why we find that @-mention is generally used to @ the internal project members.

After analyzing the usage of @-mention, we further find that issues with @-mention may have more comments, more participants, longer time-to-solve than issues without @-mention, which indicates that @-mention is very likely to be used in the difficult issues. We use the number of participants to measure the issue's difficulty and find that @-mention can reduce the time-to-solve and time delay between comments in the case of same difficulty of issues. We explain that @-mention can enlarge the visibility of issues and facilitate the developers' discussion. We also find that issues with @-mention are more likely to have label, milestone and assignee than issues without @-mention, which proves that @-mention help involve more participants and has a positive impact on the solving process of issues. Since @-mention can be used in so many ways, *e.g.* expression of disagreement or notification, we cannot be sure, without further analysis, whether an @-mention expresses trust, distrust, or none of them. We need take more research to in-depth analyze the @-mention in GitHub's issues.

During our investigation, we believe that @-mention has many possible research directions. In particular, @-mention builds a social network among the developers in the solving process of issues, which contains rich information for mining and research. Therefore, we would like to dive deep down into the data to learn more about how to use @-mention for mining developers' relationships and characteristics in GitHub.

In our preliminary work, we aim to understand the social relationships of developers in Rails project. We firstly construct a social network based on our @-mention database, which we called *@-network*. The *@-network* can be defined as a directed graph $G = (V, E)$. V represents the set of vertices which are all developers participate in Rails project. E presents a set of node pairs $E(V) = \{(u, v)|u, v \in V\}$. If the node v_j is @-mentioned by v_i, then there is a edge from v_i to v_j. For node v_i, the number of edges pointing to it is called the *indegree* $deg^-(v_i)$ and the number of edges starting from it represents its *outdegree* $deg^+(v_i)$. And the *degree* $deg(v_i)$ is the sum of *indegree* and *outdegree*. Figure 9 shows the *@-network* redrawn by *Force Atlas*[7] algorithm. The more time a developer be @-mentioned, the larger his node be.

This *@-network* graph consists of 5059 nodes and 17706 edges. We compute the *degree*, *indegree* and *outdegree* of each node. As shown in Figure 10, we find that all three degrees distribution follow a long tail distribution [15]. The average *degree* computed in the *@-network* is 3.5 (*indegree*: 1.7, *outdegree*: 1.8), revealing that each developer may @-

[7]http://gephi.org/2011/forceatlas2-the-new-version-of-our-home-brew-layout

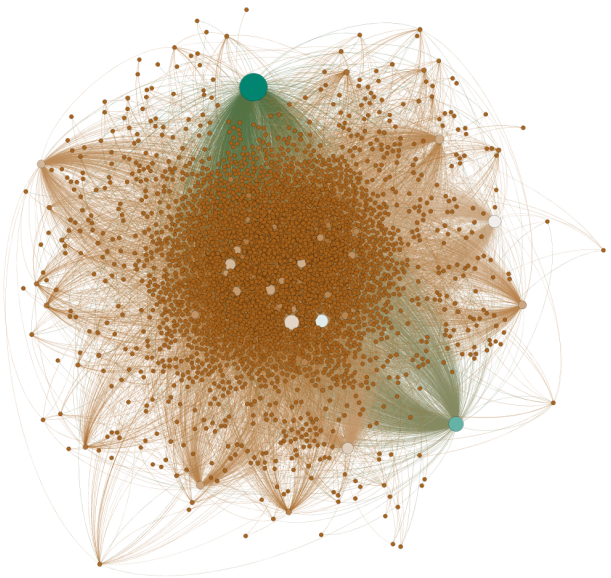
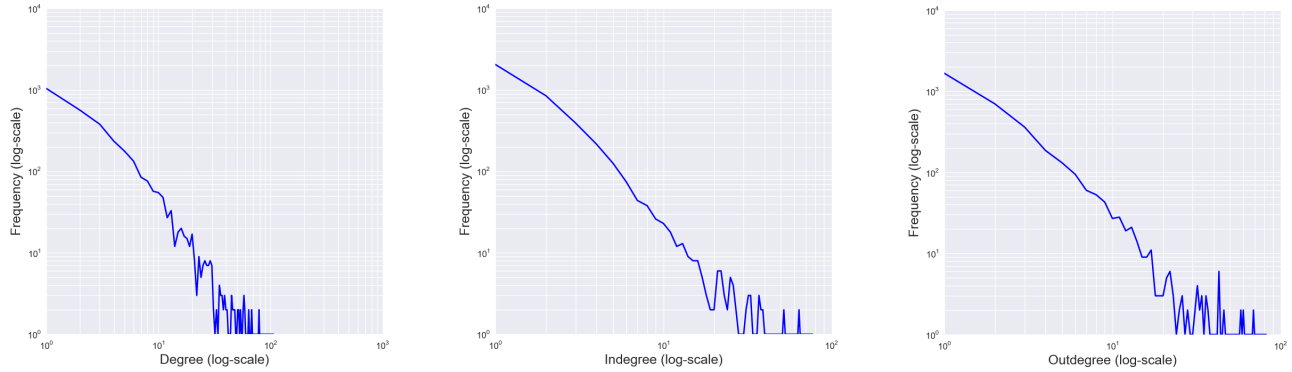


Figure 9: The @-network redrawn by Force Atlas algorithm



(a) frequency∼*degree* (b) frequency∼*indegree* (c) frequency∼*outdegree*

Figure 10: Developer degree distribution: y-axis corresponds to the number of developers having a given degree

Table 4: The top-10 influential developers in Rails

No.	id	developer	followers	starred	#commits	#comments	#general-issues	#pull-requests	#issues	page_rank_value
1	7468	rafaelfranca	858	89	2226	12020	9	107	126	0.041331
2	8994	tenderlove	>4400	221	2106	1971	7	4	11	0.018902
3	8261	senny	307	111	923	5352	8	221	229	0.018666
4	1438	carlosantoniodasilva	592	96	696	4482	1	93	94	0.015479
5	7263	pixeltrix	154	152	223	2184	16	6	22	0.015224
6	8723	steveklabnik	>2300	341	88	2983	5	63	68	0.012908
7	4635	josevalim	>4000	223	1123	2874	7	6	13	0.012342
8	4289	jeremy	745	272	1152	1899	4	9	13	0.010521
9	8308	sgrif	164	6	481	2269	3	243	246	0.009154
10	3183	fxn	520	48	795	1998	4	2	6	0.008907

mention 1.8 other developers and be @-mentioned 1.7 times. By following Surian *et al.* [16] and Leskovec *et al.* [17], the shortest path between two nodes is computed by ignoring the weights of the edges in the graph. The length of a path between two nodes is simply the length of the series of nodes between the two nodes. In our *@-network*, the diameter of the largest connected component is 10 and the average shortest path is 3.19. Surian *et al.* [16] studied the Sourceforge[8] project hosting platform and found that the average shortest path among project developers is 6.55, following the popular assumption of "six-degree-separation" [18]. Other study of the Facebook social graph has concluded that individuals on Facebook have potentially tremendous reach with an average shortest path of 4.7 [19]. The average shortest path in *@-network* is significantly lower, which suggests that the social media tool @-mention actually enables more collaboration among developers. The *@-network* allows for even better reach as developer's relationships are tighter than human's relationships in daily life social networks.

Furthermore, we want to identify the influential developers from the *@-network*. We run the PageRank algorithm in the *@-network*. PageRank algorithm, which is for weighting web pages importance based on their links, has gained popularity driven by its use in the Google search engine [20]. In our study, we consider each developer as a web page and @-mention as the URL link. There are many interactions in our PageRank algorithm. In the initial interaction, the algorithm assigns the same PageRank score to all developers. Then subsequent interactions update these scores: the score of a developer d is distributed to the developers that d @-mention to; each @-mentioned developer receive $\frac{1}{|L_d|}$ of the score, where L_d is the set of developers that d @-mention to. The PageRank score of a developer d at iteration i can be computed by the following equation. Where r represents the damping factor), T is the number of developers in our database, K_d is the set of developers that @-mention d, and L_q is the set of developers that q @-mention to.

$$PR(d,i) = \frac{1-r}{T} + r \sum_{q \in K_d} \frac{PR(q,i-1)}{|L_q|}$$

The PageRank algorithm returns a PageRank score for every developer. We can get the top-10 influential developers in terms of their PageRank scores which are shown in Table 4. The top-1 developer is "rafaelfranca", who is one of the internal project members of Rails. This developer submitted 2226 commits and 12020 comments, which indicates that he is an active developer and has a lot of contributions to the Rails. The top-2 developer is "tenderlove", who is the internal project member too. He has more than 4400 followers which reveals that he is a famous and influential developer in GitHub. Analyzing these PageRank scores and characteristics of developers, we find that the characteristics of developers are basically consistent with the PageRank scores, *i.e.* we can evaluate the contribution, activeness and influence of developers by mining the *@-network*. Nevertheless, more research needs to be done to confirm and expand these preliminary results.

[8]http://sourceforge.net/

5. RELATED WORK

Social media tools are widely used in the software development context. These social media tools leverage articulated social networks and observed code-related activity simultaneously to support the type of awareness that only available to core developers in previous [2]. In order to enhance the collaboration in software development, some research proposed the tagging [21], searchable graphs of heuristically linked artifacts [22], and workspace awareness [23] to support the coordination.

Storey M A *et al.* [24] investigated the benefits, risks and limitations of using social media in software development at the team, project and community levels. Julia Kotlarsky *et al.* [25] proposed that social ties and knowledge contribute to successful collaboration in globally distributed information system development teams. In their study, they made the point that human-related issues involving rapport and transactive memory were important for collaborative work in the software development. Black S *et al.* [26] described the preliminary results of a pilot survey conducted to collect information on social media use in global software systems development and find that social media can enable better communication through the software system development process. In particular, their research results showed that 91% of respondents said that the social media has improve their working life.

As mentioned from O'reilly T [27], social media tools can be characterized by an underlying "architecture of participation" that supports crowdsourcing as well as a many-to-many broadcast mechanism. Ahmadi *et al.* [28] found that today's generation of developers frequently makes use of social media, to augment tools in their development environments. Park S *et al.* [6] proved that blogs are frequently used by developers to document "how-to" information, to discuss the release of new features and to support requirements engineering. Louridas P *et al.* [5] proposed that wikis are used to support defect tracking, documentation, requirements tracking, test case management and for the creation of project portals. Riemer K *et al.* [7] argued that decision makers should vest trust in their employees in putting microblogging to productive use in their group work environments. Basically, these researches mainly focused on the correlation between the general social media and the overall software development. Our work is focused on analyzing how the special @-mention tool influence the issues of projects in GitHub.

@-mention, a typical social media tool used in social networking websites, *e.g.* Facebook and Twitter, allows users to reference a specific user by simply placing an "@" symbol in front of the username they wish to reference [4]. Yang J *et al.* [9] found that @-mention is a strong predictor of information diffusion. Lumbreras A *et al.* [29] proposed that @-mention usually express some kind of close or familiar relationships and can be treated as a positive indicator of mutual trust. The study presented by Vega *et al.* [10] reported that @-mention is a significant factor in enlarging the visibility of a post and helping initiate response and conversations. We had a primary investigation of @-mention used in the pull-requests hosted in Ruby on Rails [3]. Extending this prior work, we also conducted an exploratory study of @-mention in pull-requests based software development, including its current situation and benefits [30]. We have proved that @-mention is beneficial to the processing of pull-requests in GitHub. We believe it would be interesting to expand this work by considering the @-mention used in total issues.

6. THREATS TO VALIDITY

Our statistical analysis mainly use the number of comments, the number of participants *etc.* as measurements to verify the impact of @-mention on the issues. Future work is needed on analyzing some other characteristics of issues, *e.g.* the number of files changed and the code churn in *pull-requests*. In this study, we only study the issues of Ruby on Rails project in GitHub. In the future, we plan to mitigate this threat further by including more projects.

7. CONCLUSIONS

In this paper, we obtain a deep understanding of how @-mention is used in the issues of GitHub, including its usage and influence. By statistical analysis, we find that @-mention, as a famous social media widely used in online social platform, is used in Rails project quite a lot too, especially in the *pull-requests*. Most @-mention are used in the issues' comments for submitters @-mentioning other reviewers and most @-mentioned developers are internal project members. Furthermore, we find that issues with @-mention are likely to have more comments, more participants as well as longer time-to-solve than issues without @-mention, which proves that @-mention is beneficial for involving more collaboration and tends to be used in the difficult issues. While in the same difficulty of issues, @-mention can reduce the time delay during the conversation of developers because @-mention enlarges the visibility of issues and facilitates the developers' discussion. And issues with @-mention are more likely to have label, milestone as well as assignee than issues without @-mention. Our investigation shows that @-mention has a positive impact on the solving process of issues. Based on the @-mention database, we also build a *@-network* for mining the relationships and characteristics of developers in Rails.

Next steps will focus on expanding our investigation around the study of *@-network*. We plan to extend our approach to more projects in GitHub to see if there are significant differences across them.

8. ACKNOWLEDGMENTS

The research is supported by the National Natural Science Foundation of China (Grant No.61432020, 61472430 and 61502512) and the Postgraduate Innovation Fund (Grant No.CX2015B028).

9. REFERENCES

[1] Tsay J, Dabbish L, Herbsleb J D. Social media in transparent work environments. In *Proceedings of the 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, pages 65-72. IEEE, 2013.

[2] Dabbish L, Stuart C, Tsay J, et al. Social coding in GitHub: transparency and collaboration in an open software repository. In *Proceedings of the Conference on Computer Supported Cooperative Work*, pages 1277-1286. ACM, 2012.

[3] Zhang Y, Yin G, Yu Y, et al. Investigating social media in GitHub's pull-requests: a case study on Ruby on

Rails. In *Proceedings of the 1st International Workshop on Crowd-based Software Development Methods and Technologies*, pages 37-41. ACM, 2014.

[4] Meeder B, Tam J, Kelley P G, et al. RT@ IWantPrivacy: Widespread violation of privacy settings in the Twitter social network. In *Proceedings of the Web*, 2, pages 1-2. 2010.

[5] Louridas P. Using wikis in software development. *Software*, 23(2), pages 88-91. IEEE, 2006.

[6] Park S, Maurer F. The role of blogging in generating a software product vision. In *Proceedings of the 2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering*, pages 74-77. IEEE, 2009.

[7] Riemer K, Richter A. Tweet inside: Microblogging in a corporate context. In *Proceedings of the 23rd Bled eConference*, pages 1-17. 2010.

[8] Begel A, DeLine R, Zimmermann T. Social media for software engineering. In *Proceedings of the FSE/SDP workshop on Future of software engineering research*, pages 33-38. ACM, 2010.

[9] Yang J, Counts S. Predicting the Speed, Scale, and Range of Information Diffusion in Twitter. In *Proceedings of ICWSM*, pages 355-358. 2010.

[10] Vega E, Parthasarathy R, Torres J. Where are my tweeps?: Twitter usage at conferences. *Paper, Personal Information*, pages 1-6. 2010.

[11] Bird C, Gourley A, Devanbu P, et al. Open borders? immigration in open source projects. In *Proceedings of the 4th International Workshop on Mining Software Repositories*, pages 6-6. IEEE, 2007.

[12] Cabot J, Canovas Izquierdo J L, Cosentino V, et al. Exploring the use of labels to categorize issues in Open-Source Software projects. In *Proceedings of the 22nd International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 550-554. IEEE, 2015.

[13] Thung F, BissyandÍę T F, Lo D, et al. Network structure of social coding in github. In *Proceedings of the 17th European Conference on Software Maintenance and Reengineering (CSMR)*, pages 323-326. IEEE, 2013.

[14] Gousios G, Pinzger M, and van Deursen A. An exploration of the pull-based software development model. In *Proceedings of the 36th International Conference on Software Engineering (ICSE)*, 2014.

[15] Anderson C. The long tail: how endless choice is creating unlimited demand. *Random House*, 2007.

[16] Surian D, Lo D, Lim E P. Mining collaboration patterns from a large developer network. In *Proceedings of the 17th Working Conference on Reverse Engineering (WCRE)*, pages 269-273. IEEE, 2010.

[17] Leskovec J, Horvitz E. Planetary-scale views on a large instant-messaging network. In *Proceedings of the 17th International Conference on World Wide Web*, pages 915-924. ACM, 2008.

[18] Travers J, Milgram S. An experimental study of the small world problem. *Sociometry*, pages 425-443. 1969.

[19] Ugander J, Karrer B, Backstrom L, et al. The anatomy of the facebook social graph. *arXiv preprint* pages 1111-4503 arXiv, 2011.

[20] Brin S, Page L. Reprint of: The anatomy of a large-scale hypertextual web search engine. *Computer networks*, 56(18), pages 3825-3833. 2012.

[21] Storey M A, Ryall J, Singer J, et al. How software developers use tagging to support reminding and refinding. *IEEE Transactions on Software Engineering*, 35(4), pages 470-483. IEEE, 2009.

[22] Froehlich J, Dourish P. Unifying artifacts and activities in a visual tool for distributed software development teams. In *Proceedings of the 26th International Conference on Software Engineering*, pages 387-396. IEEE, 2004.

[23] Omoronyia I, Ferguson J, Roper M, et al. Using developer activity data to enhance awareness during collaborative software development. *Computer Supported Cooperative Work (CSCW)*, 18(5-6), pages 509-558. 2009.

[24] Storey M A, Treude C, van Deursen A, et al. The impact of social media on software engineering practices and tools. In *Proceedings of the FSE/SDP workshop on Future of software engineering research*, pages 359-364. ACM, 2010.

[25] Kotlarsky J, Oshri I. Social ties, knowledge sharing and successful collaboration in globally distributed system development projects. *European Journal of Information Systems*, 14(1), pages 37-48. 2005.

[26] Black S, Harrison R, Baldwin M. A survey of social media use in software systems development. In *Proceedings of the 1st Workshop on Web 2.0 for Software Engineering*, pages 1-5. ACM, 2010.

[27] O'reilly T. What is Web 2.0: Design patterns and business models for the next generation of software. *Communications & strategies*, 65(1), pages 17-37. 2007.

[28] Ahmadi N, Jazayeri M, Lelli F, et al. A survey of social software engineering. In *Proceedings of Automated Software Engineering-Workshops*, pages 1-12. IEEE, 2008.

[29] Lumbreras A, Gavalda R. Applying trust metrics based on user interactions to recommendation in social networks. In *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*, pages 1159-1164. IEEE, 2012.

[30] Zhang Y, Yin G, Yu Y, et al. A Exploratory Study of @-Mention in GitHub's Pull-Requests. In *Proceedings of the 21st Asia-Pacific Software Engineering Conference*, pages 343-350. IEEE, 2014.