A Exploratory Study of @-mention in GitHub's Pull-requests

Yang Zhang, Gang Yin, Yue Yu, Huaimin Wang
National laboratory for Parallel and Distributed Processing, National University of Defense Technology
Changsha, 410073, China
zhangyanggfkd@gmail.com, jack_nudt@163.com, yuyue_whu@foxmail.com, whm_w@163.com

***Abstract*—Pull-request mechanism is an outstanding social development method in GitHub. @-mention is a social media tool that deeply integrated with pull-request mechanism. Recently, many research results show that social media tools can promote the collaborative software development, but few work focuses on the impacts of @-mention. In this paper, we conduct an exploratory study of @-mention in pull-request based software development, including its current situation and benefits. We obtain some interesting findings which indicate that @-mention is beneficial to the processing of pull-request. Our work also proposes some possible research directions and problems of the @-mention. It helps the developers and researchers notice the significance of @-mention in the pull-request based software development.**

Keywords—pull-request; social media; @-mention

I. Introduction

Pull-request as implemented by GitHub[1] in particular, is a new model for collaborating on distributed software development [1]. It attracts more and more external developers to contribute their code and suggestions to core developers. GitHub is a social collaborative software development community. The platform integrates many social media tools involving follow [2], watch [2], comment action [3] and @-mention. It upgrades the pull-request to a socialized development model.

@-mention allows developers to reference a specific user in the pull-requests by simply placing a "@" 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 pull-request's description body or the pull-request's comments, which makes it more deeply involved in the processing of pull-request.

Previous work has identified the impact of social media on software development. They found that social media plays an increasingly important role in software engineering research and practice [8]. It has changed the way that people collaborate and share information [9]. In addition, social media could enable better communication through the process of the software system development [10]. Basically, these researches mainly focused on the correlation between the general social media and the overall software development. @-mention has been proved as a significant factor in enlarging the visibility of a post and helping initiate responses and conversations [11]. However, for the current situation and benefits of @-mention in GitHub's pull-requests, we still lack a comprehensive understanding, which makes the work still at the very beginning. There are many concerns reserved, for instance, how popular is @-mention in the pull-requests? what kind of differences in the complexity between the pull-requests with @-mention and the pull-requests without @-mention? to what extent does @-mention support developers' collaboration in the pull-requests?

[1]Https://github.com/

In this paper, we conduct an exploratory study of @-mention in GitHub's pull-requests. By using the qualitative and quantitative approaches, we obtain some insights of @-mention in the pull-request based software development. Our results indicate that, @-mention plays an important role in facilitating the developers' collaboration by reducing the delay time in the processing of pull-request. However, we find that @-mention is not widely used in the pull-requests, and the current mechanisms in @-mention do not visibly improve the productivity of the collaborative development. For instance, we find that developers could not @ the suitable developers effectively and easily when they are unfamiliar with each other. Based on the study, we propose some possible research directions of @-mention which might be worth being invested to improve the pull-request based software development.

In summary, our main contributions in this paper include:

1) To the best of our knowledge, we are the **first** to give a quantitative and qualitative study on @-mention in pull-requests. This study gives some important implications for the developers to make better use of @-mention in GitHub.

2) We analyze the correlation between the specific location of @-mention and the cost time in the processing of pull-request. Our observation can be used to guide the software developers to use the @-mention in the right location.

3) We propose some promising research directions and problems, which would guide future software engineering tool innovations as well as practices.

The remainder of this paper is structured as follows. Section II describes related work. In Section III, we introduce the related concepts and our research questions. Section IV presents our empirical study methodology, and Section V presents results of the study. We discuss findings based on the study in Section VI. Threats to validity are discussed in Section VII. We conclude the article in Section VIII.

II. Related work

In earlier studies, there are many socially related technologies used in the software development context. Social technologies 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. In order to enhance the collaboration in software development, some research proposed the tagging [12], searchable graphs of heuristically linked artifacts [13], and workspace awareness [14] to support the coordination. What's more, Louridas P [5] find that wikis are used to support defect tracking, documentation, requirements tracking, test case management and for the creation of project portals. 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. Riemer K and Richter A [7] argued that decision makers should vest trust in their employees in putting microblogging to productive use in their group work environments. Ahmadi et al. [15] find that today's generation of developers frequently makes use of social media, to augment tools in their development environments. As mentioned from O'reilly T [16], social media tools can be characterized by an underlying "architecture of participation" that supports crowdsourcing as well as a many-to-many broadcast mechanism. Storey M A et al. [8] investigated the benefits, risks and limitations of using social media in software development at the team, project and community levels. Julia Kotlarsky et al. [17] find 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. [10] 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.

These previous researches basically focused on the correlation between the general social media and the overall software development. Although Yang et al. [18] have a primary investigation of @-mention in the Ruby on Rails, we are not aware of any in-depth empirical study dedicated in such claim. It inspires us to put forward a comprehensive analysis of @-mention in the pull-requests of GitHub.

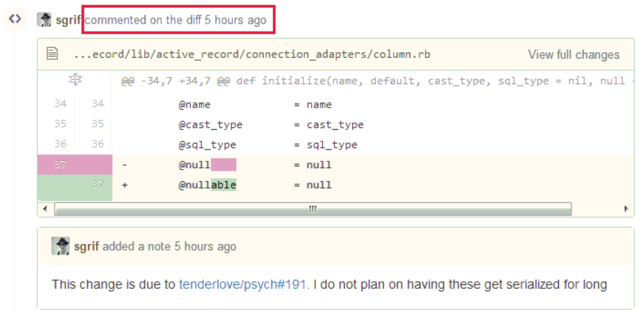
III. Preliminaries & Problem Definition

In this section, we briefly introduce the @-mention and propose our research questions.

A. @-mention

@[2], normally read as "at", is the meaning of "located at" or "directed at", especially in email addresses such as tom@example.com (the tom located at site the example.com). In recent year, more and more online social media, without threaded discussions, use @ to denote a reference or a reply, such as Facebook[3], Twitter[4]. The feature of @-mention enables users to directly reference others by putting a "@" symbol before their username such as "@Tom". Then @-mention can automatically interpret these as links to the user's

[2]Http://en.wikipedia.org/wiki/@
[3]Https://www.facebook.com/
[4]Https://www.twitter.com/



sgrif commented on the diff 5 hours ago

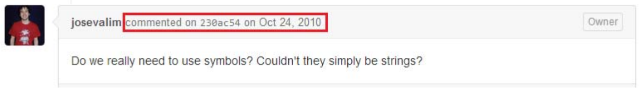
...ecord/lib/active_record/connection_adapters/column.rb View full changes

```
@@ -34,7 +34,7 @@ def initialize(name, default, cast_type, sql_type = nil, null
        @name           = name
        @cast_type      = cast_type
        @sql_type       = sql_type
-       @null           = null
+       @nullable       = null
```

sgrif added a note 5 hours ago

This change is due to tenderlove/psych#191. I do not plan on having these get serialized for long

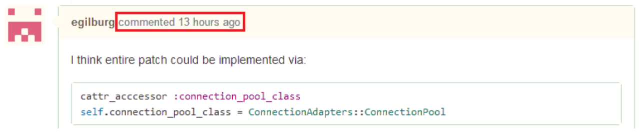
(a) pull-request review comments



josevalim commented on 230ac54 on Oct 24, 2010 Owner

Do we really need to use symbols? Couldn't they simply be strings?

(b) commit comments



egilburg commented 13 hours ago

I think entire patch could be implemented via:

```
cattr_acccessor :connection_pool_class
self.connection_pool_class = ConnectionAdapters::ConnectionPool
```

(c) issue comments

Fig. 1. Three type of comments in the pull-requests

profile. In addition to the link function, after @-mentioning somebody, the @-mentioned person could receive a reminder to help himself respond immediately. In the pull-requests of GitHub, there are three locations of @-mention: pull-request' title, pull-request's description body (pull-request's body) and pull-request's comments. It should be mentioned that @-mention in the pull-request's title does not have the link function because it is just a text, not like the others. So in our study, we only discuss the @-mention that used in the body and in the comments.

In fact, there are three type of comments in the pull-requests: pull-request review comments, commit comments and issue comments. Pull-request review comments are comments on a portion of the diff patch like Fig.1(a). They are applied directly to the detailed modification. Commit comments are comments on a commit like Fig.1(b). They are out of the pull-request review comments. Issue comments are comments on a pull-request itself like Fig.1(c). They are out of the commit comments.

B. Research Questions

In order to have a detailed research, we formulate research questions within three categories: quantity related questions, quality related questions and effect related questions.

RQ1: To what extent is @-mention used in the pull-request paradigm?

In answer to this research question, we investigate the distribution of @-mention in different locations and different scenarios. Then we statistic the utilization of @-mention in different type of pull-requests.

RQ2: What kind of differences are there between the pull-requests with and without @-mention?

In this question, we mainly focus on the following characteristics of pull-requests: the number of commits, the number of comments, and the number of participants (pull-requests' submitters and comments' submitters), as well as the handling time.

RQ3: How well do @-mention in the pull-requests support the developers for communication? Does the specific location of @-mention impact the processing of pull-request?

For answering the questions, we give a detailed investigation about the impact of @-mention, as well as its specific location on different cost time in the processing of pull-request.

IV. Methodology

In this section, we give our datasets, the preprocessing and the statistical measurements.

A. Datasets

Our empirical study is based on the combination of two famous datasets: GitHub Archive and GHTorrent.

GitHub Archive[5], *githubarchive.org*, records the public developing activities on the Git repositories stored on GitHub. The activities are aggregated in hourly archives, providing 18 events types involving new commits, fork, commenting, adding members etc. The Archive encodes these events into a Json file. The contents of the file consist of Json objects appended one after the each. During our work, we download and parse the Archive data from January 2013 to March 2014. The full volume of these Archive data is approximately 160GB representing over 100 million events.

GHTorrent[6], *ghtorrent.org*, is a scalable, offline mirror of the data offered through the GitHub Rest API [19]. The GHTorrent already offer data dumps of both its raw data that stored in MongoDB (currently more than 2TB), and metadata that stored in MySQL (currently more than 20GB) [20]. After downloading and parsing the database dump, we can get almost all of the development information of the repositories in the GitHub like commits, issues, pull-requests, projects and users. This process is very time consuming.

B. Preprocessing

In order to extract the information of @-mention from the comments (pull-request review comments, commit comments and issue comments), we need to preprocess our datasets. In the GHTorrent, issue comments' description bodies and pull-requests' bodies are missing. However, in the GitHub Archive, the body information is not lost. Fig.2 shows our approaches for solving the missing problem. First, we extract the basic information of pull-requests and comments from GHTorrent to build the basic *PR dataset*. Then, we extract the missing body information from GitHub Archive to build the additional *Body dataset*. Finally, we combine the two datasets with the unique pull-request ID to build our final *Research dataset*. In this dataset, the missing information of pull-requests is complete. This is a tough and tedious process, which might be one reason for there are few researches that investigated the @-mention in the pull-requests.

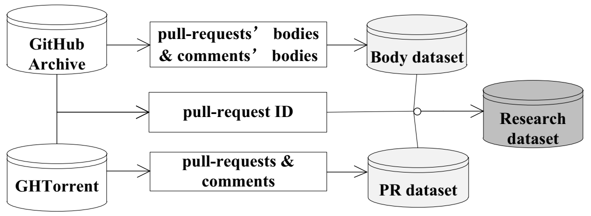


Fig. 2. Overview of our final dataset forming approach

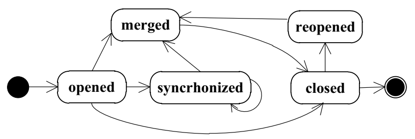


Fig. 3. Different states in a pull-request's lifecycle

For avoiding disturbances, we mainly analyze the data of 3587 projects (not deleted). These projects contain at least 100 pull-requests. They cover 53 different program languages. As shown in Table I, Top-5 program languages contain 67% of projects. Where the symbol # indicates counting.

TABLE I. The Top-5 program languages

Language	JavaScript	Ruby	Python	Java	PHP
#Project	678	475	460	424	362

As shown in Fig.3, in a pull-request's lifecycle, there are five states: *opened*, *synchronized*, *merged*, *closed* and *reopened*. When a pull-request's processing is finished, it must be closed but not necessarily be merged. Besides, in our study, we consider that reopening a closed pull-request should not be considered, since it is not a common practice and it is not easily detectable. So we only focus on the processing of a pull-request from its *opened* state to its first *closed* state. Because of the above reasons, in our dataset, we select the 744684 closed pull-requests out of the total 1038117 pull-requests. After removing the invalid data such as comment-timestamp earlier than open-timestamp or open-timestamp later than closed-timestamp, we obtain 724623 pull-requests. Because there are some projects, all of their pull-requests are submitted by one type of developers: core developers or external developers. We filter out these unbalanced projects. Finally, we collect 566538 closed pull-requests from 2006 projects for the later analysis and research.

C. Statistical Measurements

In our study, the statistical measurements include:

1) Cost time in pull-requests

During a typical pull-request's processing, we divide the cost time into 3 parts:

a)Time To Handle (TTH): The time interval between a pull-request is opened and closed. The follows is defined for a generic pull-requests *p*.

$$TTH(p) = timestamp_c(p) - timestamp_o(p) \quad (1)$$

b)Delay Before Comment (DBC): The time interval between a pull-request is opened and it receives the first comment.

$$DBC(p) = timestamp_{first-c}(p) - timestamp_o(p) \quad (2)$$

[5]Http://www.githubarchive.org/
[6]Http://ghtorrent.org/

c)Delay In Comments (DIC): The time interval between a comment *x* and its next comment *y*.

$$DIC(p) = timestamp_{c-y}(p) - timestamp_{c-x}(p) \quad (3)$$

2) Statistics tests

In the analysis of statistical significance between the distributions of pull-requests with @-mention and without @-mention, we use Mann-Whiney-Wilcoxon test, Z test and Cliff's δ. Those statistics tests are non-parametric statistical hypothesis tests. They do not assume any specific distribution, which is a suitable property for our experimental analysis.

a)Mann-Whiney-Wilcoxon (MWW) test: In the MWW test, there are two independent samples *X* and *Y*, of size N_1 and N_2 respectively. First, the two samples are combined into an ascending order list where data points with identical values are assigned the same rank. Then, the test sums the ranks of data points in the first sample *X*. We denote this sum as *T*. By using the MWW test, we can evaluate whether these samples are drawn from the same distribution. The following is the formula for computing Mann-Whitney *U* for *X*:

$$U = N_1 N_2 + \frac{N_1(N_1+1)}{2} - T \quad (4)$$

Where *U* is computed to determine the *p*-value. If the significance level α is 0.001, *p*-value less than α means the test rejects the null hypothesis, which verifies that the two samples have different distributions at the significance level of 0.001.

b)Z test: Z test is generally used for comparing the mean difference of large samples (size > 30). Considering the two samples *X* and *Y* described above, their mean values are denoted as $\bar{X}$ and $\bar{Y}$, their standard deviation values are denoted as S_1 and S_2. *Z* is calculated by the following formula:

$$Z = \frac{\bar{X}-\bar{Y}}{\sqrt{{S_1}^2/N_1 + {S_2}^2/N_2}} \quad (5)$$

Where $|Z| \geq 2.58$ means the difference is "very significant", $|Z| \geq 1.96$ means "significant", $|Z| < 1.96$ means "not significant".

c)Cliff's δ: Cliff's δ is a non-parametric effect size measure that quantifies the amount of difference between two samples. The following is the formula for computing δ for *X* and *Y* described above:

$$\delta = \frac{\#(X>Y) - \#(X<Y)}{N_1 N_2} \quad (6)$$

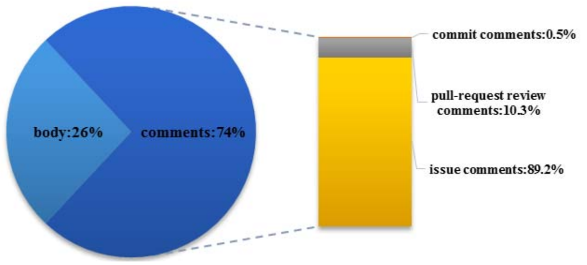
Where $|\delta| < 0.147$ means the difference is "negligible", $|\delta| < 0.33$ means "small", $|\delta| < 0.474$ means "medium", otherwise means "large".

V. Empirical Evaluation

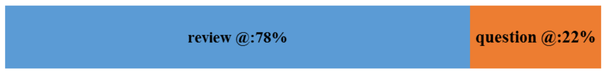
In this section, we present the results of our empirical study. These results are reported as responses to the research questions that were provided in Section III-B.

A. RQ1: Current situation of @-mention

As mentioned in Section III-A, @-mention is usually used in the pull-request's body, pull-request review comments, commit comments or issue comments. As shown in Fig.4(a),



(a) Specific location of @-mention



(b) Specific scenario of @-mention

Fig. 4. Distribution of @-mention in the pull-requests

26% @-mention come from the pull-requests' bodies. 74% @-mention come from the pull-requests' comments: issue comments (89.2%), pull-requests review comments (10.3%), and commit comments (0.5%). We divide the scenarios of @-mention into two kinds: "review @" (the pull-requests' submitters @ other developers for review) and "question @" (the comments' submitters @ the pull-requests' submitters for question). Fig.4(b) shows that 78% @-mention are used for review. This indicates that ***most @-mention are used by the pull-requests' submitters for review***. ***They are mainly used in the pull-request's body and the issue comments***.

Then, we discuss the utilization of @-mention by considering two type of pull-request contributions: internal contributions and external contributions. Internal contributions are pull-requests submitted by core developers (project members). External contributions are pull-requests submitted by external developers (not project members). It should be noted that in our study, we consider that @-mention is used in a pull-request, as long as the pull-request has one or more valid @-mention operation. Similarly, we discuss the percentage of @-mention used in the comments, as long as one of the three type of comments has @-mention.

TABLE II. Utilization of @-mention in different contributions

	Total		Internal		External	
	#PR	**Ratio**	**#PR**	**Ratio**	**#PR**	**Ratio**
have comments	245989	43.4%	106296	39.0%	139693	47.6%
@ in comments	53950	21.9%	24438	23.0%	29512	21.1%
have body	395384	69.8%	177317	65.0%	218067	74.3%
@ in body	19713	5.0%	12779	7.2%	6934	3.2%
@ in total	68625	12.1%	33971	12.5%	34654	11.8%

Based on our statistics, there are 272847 internal contributions and 293691 external contributions. Table II shows the utilization of @-mention in different contributions. In total, only 12.1% pull-requests have @-mention. This indicates that ***@-mention is not widely used in the pull-requests***.

In our opinion, the description body is better reflecting the purpose of a pull-request. But from our results, we find that the percentage of @-mention used in the body is 5.0%, less than the value of @-mention used in the comments (nearly 22.0%). The current @-mention is manually used by developers. So we assume that if the pull-request's submitter is unfamiliar with the suitable reviewers, it is difficult to decide to @ whom at the time of the pull-request be submitted.

While, along with the discussion, the other participants may help solve this "@ whom" problem. In our study, we find that 60.2% @-mentioned developers come from the internal group. In particular, this effect is more remarkable among the external contributions (nearly 79.0%). It proves that @-mention is generally used to @ the internal developers. So it is easier for the internal developers to @ the suitable reviewers than the external developers because these internal developers are familiar with each other. This indicates that ***the current mechanisms in @-mention bring some usage problems to the developers, especially in the external contributions***.

RQ1: @-mention is not widely used in the pull-requests. The utilization of @-mention indicates that there may be some weakness of the current mechanisms in @-mention, especially in the external contributions.

B. RQ2: The characteristics of pull-requests with @-mention

In this investigation, we mainly focus on the following characteristics of pull-requests: the number of commits, the number of comments, the number of participants and the handling time (TTH). In order to give a clear description of our results, we use the R statistical analysis boxplots tool in our study. 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).

1) The number of commits

Fig.5 shows the distribution of the number of commits in pull-requests with @-mention and without @-mention. The average number of commits is 4.1 (median: 1.0) for pull-requests without @-mention, while the number is raised to 5.4 (median: 2.0) for pull-requests with @-mention. Using the statistical test, we verify that the difference between the two groups is statistically significant (p<0.001, z=17.8, δ=0.17). This indicates that ***pull-requests with @-mention is more likely to have more commits***.

2) The number of comments

Fig.6 shows the distribution of the number of comments in pull-requests with @-mention and without @-mention. The average number of comments is 1.0 (median: 0) for pull-requests without @-mention, while the number is raised to 7.2 (median: 4.0) for pull-requests with @-mention. We test and confirm that the two distributions are significantly different using the statistical test (p<0.001, z=131.9, δ=0.71). This indicates that ***pull-requests with @-mention is more likely to have more comments***.

3) The number of participants

Fig.7 show the distribution of the number of participants in pull-requests with @-mention and without @-mention. The average number of participants is 1.4 (median: 1.0) for pull-requests without @-mention, while the number is raised to 2.8 (median: 2.0) for pull-requests with @-mention. In our statistical test, we test and confirm that the difference is statistically different (p<0.001, z=202.2, δ=0.64). This indicates that ***pull-requests with @-mention is more likely to have more participants***.

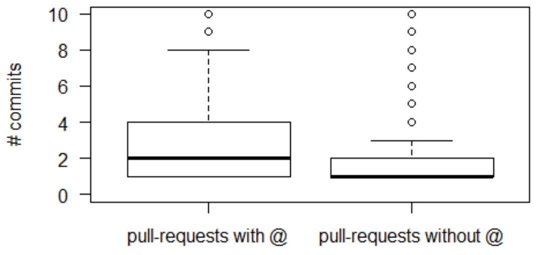


Fig. 5. Number of commits and @-mention

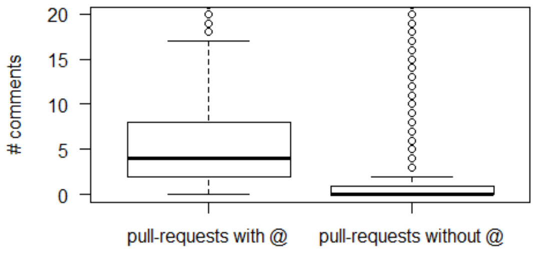


Fig. 6. Number of comments and @-mention

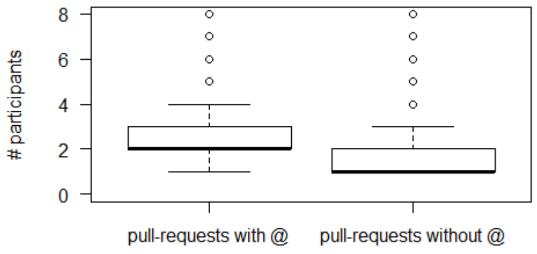


Fig. 7. Number of participants and @-mention

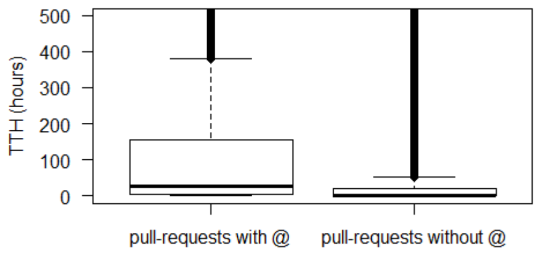


Fig. 8. TTH of pull-requests and @-mention

4) TTH

Fig.8 shows the distribution of TTH in pull-requests with @-mention and without @-mention. The average TTH is 101.5 hours (median: 2.2 hours) for pull-requests without @-mention, while the time is raised to 350.8 hours (median: 26.7 hours) for pull-requests with @-mention. The result of statistical tests reveals that the difference is statistically significant (p<0.001, z=57.2, δ=0.43). This indicates that ***pull-requests with @-mention is more likely to need more time to deal with***.

In order to further investigate the impact of @-mention on the TTH, we statistic the percentage of pull-requests with @-mention and without @-mention in different TTH. As shown in Fig.9, we find that with the growing of TTH, the percentage of pull-requests with @-mention is increasing. The gap of TTH between the pull-requests with @-mention and without @-mention is gradually reduced. If we consider the TTH as a measurement of the pull-request's complexity, it further indicates that ***@-mention is more likely to be used in the complex pull-requests***.

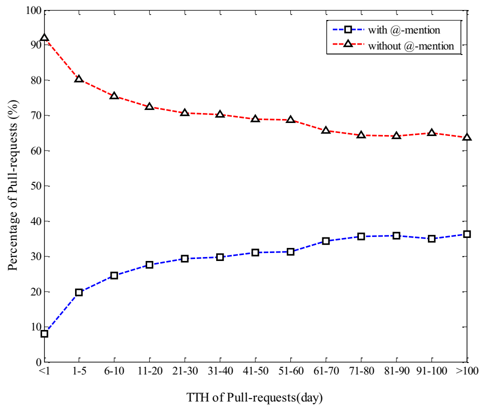


Fig. 9. Percentage of pull-requests with different TTH

RQ2: @-mention is more likely to be used in those complex pull-requests which have more commits, more comments, more participants and longer TTH.

C. RQ3: The impact of @-mention on the processing of pull-request

To utilize @-mention in the pull-requests effectively, we need to find out what effect of @-mention to the cost time in the processing of pull-request. We consider this issue from three aspects: DBC, DIC and TTH. Furthermore, we discuss the correlation between the location of @-mention and the TTH of pull-requests.

1) DBC

In this study, we only discuss @-mention which is used in the pull-request's body. We investigate whether @-mention is useful for reducing the delay-before-comment (DBC) in the processing of pull-request. We consider the first comment as the beginning of developers' collaboration. So it is better to have a DBC as shorter as possible.

As shown in Fig.10, the average DBC of pull-requests with @-mention in their bodies is 37.2 hours (median: 0.8 hours). While the average DBC of pull-requests without @-mention in their bodies is raised to 78.0 hours (median: 1.4 hours). We can find that @-mention used in the pull-request's body can reduce the delay time before the first comment. This indicates that ***@-mention used in the pull-request's body can enlarge the visibility of the pull-request. It helps the @-mentioned developers find and respond the pull-request quickly***.

2) DIC

In this study, we only discuss @-mention which is used in the pull-request's comments. We investigate whether @-mention is useful for reducing the delay-in-comment (DIC) in the processing of pull-request. A conceivable hypothesis is that @-mention used in the comments can remind the @-mentioned developers to respond immediately.

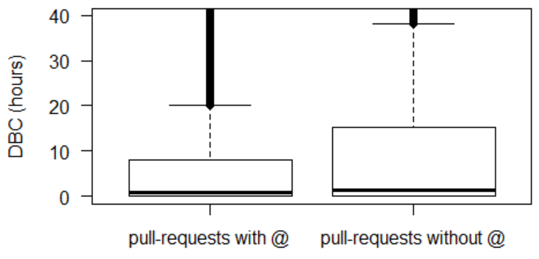


Fig. 10. DBC of pull-requests and @-mention

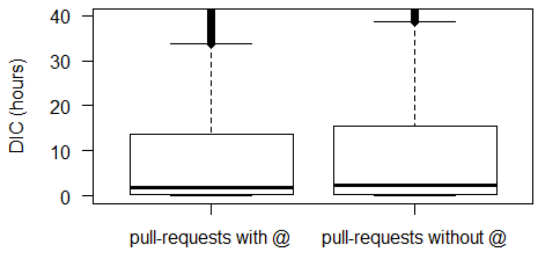


Fig. 11. DIC of pull-requests and @-mention

We collect 141484 pull-requests which have two or more comments. Based on this dataset, we find that the average DIC of pull-requests with @-mention in their comments is 57.2 hours (median: 1.8 hours). While the average DIC of pull-requests without @-mention in their comments is raised to 77.5 hours (median: 2.4 hours). Fig.11 shows the DIC of pull-requests. It shows that the pull-requests without @-mention in their comments need a little more DIC than those pull-requests with @-mention. This indicates that ***@-mention used in the comments is useful for reminding the developers to reply. It is a convenient tool in the developers' conversation.***

3) TTH

As mentioned above, there are 78% @-mention which are used for review in the pull-requests. So the number of reviewers (participants except the pull-request's submitter) is an important factor to illustrate the characteristic of pull-request's processing. In this investigation, we compare the difference of TTH between the pull-requests with @-mention and without @-mention when they have the same number of reviewers.

In fact, there are many pull-requests with a minimal TTH. Since they are closed quickly, the impact of @-mention on the TTH is hard to analyze. In addition, the percentage of @-mention usage is bigger in those complex pull-requests as outlined in RQ2. In order to reduce the interference of easier pull-requests, we only discuss the 69096 pull-requests whose TTH are more than 5 days (this percentage of pull-requests with @-mention is more than 20% as shown in Fig.9).

Based on our statistics, as shown in Table III, we find that the average TTH of pull-requests with @-mention is basically less than the value of pull-requests without @-mention. This indicates that ***considering these pull-requests reviewed by the same number of reviewers, the pull-requests with @-mention need shorter handling time***. Furthermore, with the increasing of reviewer's quantity, the gap of TTH is widening (maximum: 1398.3 hours, nearly 2 month). In particular, in the external contributions, the max gap is 2180.4 hours (nearly 3 month).

This saving time is very valuable in the current software development.

TABLE III. DIFFERENCE BETWEEN PULL-REQUESTS WITH @-MENTION AND WITHOUT @-MENTION IN DIFFERENT NUMBER OF REVIEWERS

#Reviewers	With @-mention			Without @-mention		
	#PR	Ratio	Avg. TTH(hrs)	#PR	Ratio	Avg. TTH(hrs)
0	354	0.12	592.2	2609	0.88	771.6
1	4681	0.18	833.0	20838	0.82	912.9
2	6117	0.46	1018.2	7240	0.54	1237.1
3	3916	0.65	1172.3	2082	0.35	1486.6
4	1991	0.76	1510.1	632	0.24	1665.1
5	1046	0.82	1879.4	223	0.18	2068.1
6	470	0.88	2215.1	67	0.12	1726.4
7	221	0.91	2486.1	21	0.09	3757.4
8	140	0.92	2396.1	12	0.08	3784.4

4) The correlation between the location of @-mention and the TTH

Furthermore, we analyze whether the specific location of @-mention affect the TTH of pull-requests. From our statistics, we find that the average TTH of pull-requests with @-mention only used in their bodies is 77.6 hours TTH (median: 3.3 hours). While the value is raised to 171.4 hours (median: 29.8 hours) for pull-requests with @-mention only used in the pull-request review comments. The average TTH is 450.4 hours (median: 43.6 hours) for pull-requests with @-mention only used in the issue comments. For the pull-requests with @-mention only used in the commits comments, the average TTH is 719.8 hours (median: 124.7 hours). Fig.12 shows the TTH of pull-requests with @-mention used in different places.

We find that pull-requests with @-mention used in their bodies have fewer TTH, because @-mention can accelerate the beginning of developers' collaboration as described in the discussion of DBC. And @-mention used in the pull-request review comments have more excellence to reduce the TTH compared to other comments. One explanation is that the pull-request review comments are more direct to the specific content of the pull-requests than other comments as described in Section II-B. This indicates that ***the location of @-mention has an apparent influence to the TTH of pull-requests. More direct to the specific content of pull-requests, more beneficial to the processing of pull-request***. This also guides the developers where to use the @-mention to reduce more delay time in the processing of pull-request.

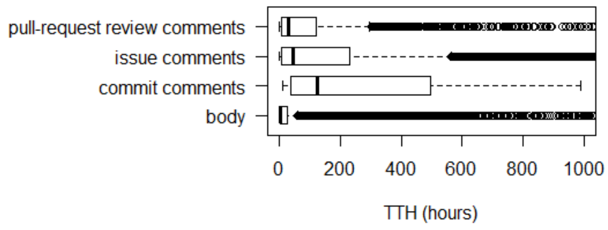


Fig. 12. Prevalence of TTH for location of @-mention

RQ3: @-mention can better support the pull-request processing by reducing the delay time in developers' collaboration. The locations of @-mention have different effect to the processing of pull-request.

VI. FINDINGS

Based on the study, we draw some findings with the expectation of supporting related researches in the future:

A. @-mention is a very useful social media tool in the pull-requests.

As the results indicate, @-mention used in a pull-request's body can enlarge the visibility of the pull-request and reduce the delay time before the first comment. @-mention used in a pull-request's comments can facilitate the developers' discussion by reducing the delay time in the comments. Furthermore, if two pull-requests reviewed by the same number of developers, the pull-request with @-mention need shorter handling time than the pull-requests without @-mention. Therefore, in GitHub, ***@-mention plays an important role in the pull-request based software development***.

B. Mechanisms need be provided to support developers use @-mention.

Our study find that @-mention is an efficient tool but not widely used in the pull-requests, especially in the external contributions. An important reason is that the current @-mention are manually used by the developers. It is difficult for the developers to @ the suitable developers when they are unfamiliar with each other. In addition, our results show that complex pull-requests are more likely to have @-mention, but the current mechanisms in @-mention do not support the development collaboration well. Therefore, ***mechanisms for helping developers use @-mention in the pull-requests easily and effectively are needed***. For instance, when a developer puts an "@" in a pull-request's comment, the platform should automatically list some suitable developers for @-mentioning.

C. @-mentin has many possible research directions.

Through our study, we find that @-mention has following meaningful research directions:

1)Knowing the development activities of developers. For instance, our investigation shows that the locations of @-mention have different effect to the pull-request's processing. The more direct to the pull-request's content, the more useful for reducing delay time. This indicates that the developers' development activities need focus on the pull-request's inside, such as a specific diff patch in a commit file.

2)Mining developers' relationship and characteristics in GitHub. @-mention builds a social network among the developers, which contain rich information for mining. By analyzing this "@ network", we can find out the developers' relationship and characteristics. For instance, the frequently @-mentioned developer may be the expert in the domain.

3)Assigning more suitable reviewers. In fact, the process of assigning pull-requests to developers is done manually by the manager. It is a time consuming process. Also the manager can only assign one developer to review the pull-request. According to our statistics, the percentage of assigned pull-requests in our datasets is just 0.89%. This indicates that it is a tedious and tough job for manager to assign the suitable reviewers. If we consider @-mention as recommending highly relevant reviewers to review the pull-requests, it is largely cutting down the workload of the project managers because more developers can help do this assigning work. Furthermore, compared to the traditional assigning method, @-mention takes more reviewers into account. It makes the reviewing work of pull-requests more efficient and time-saving.

VII. Threats to Validity

A. Internal validity

Our statistical analysis uses the number of commits etc. as measurements to verify the characteristics of pull-requests with @-mention. Future work is needed on analyzing the number of files changed and the code churn of a pull-request.

B. External validity

The abnormal pull-requests are the pull-requests that contain a few simple modifications but have a long handling time. Although we have filtered out the unbalanced projects in our datasets, the datasets still contain some projects that have some abnormal pull-requests. This may lead to bias in our survey.

VIII. Conclusions

The goal of this study is to obtain a deep understanding of @-mention in the GitHub's pull-requests, including its current situation and benefits. The study indicates that @-mention is a useful social media tool for developers' collaboration in the pull-request based software development. However, by statistical analysis, we find that the current mechanisms in @-mention do not visibly improve the productivity of the collaborative development in GitHub. More researches should be conducted to leverage the power of @-mention in the pull-request based software development. From the perspective of pull-requests' submitters (especially the external contribution's submitters), how to help them effectively find suitable reviewers by using @-mention should be investigated. According to the study results, we draw some findings with the expectation that more practices and researches being focused on this @-mention. We look forward to helping the researchers and developers understand the significance of @-mention in the pull-requests.

Acknowledgment

We thank Minghui Zhou, Charles X. Ling, and the APSEC reviewers for their valuable feedback on earlier revisions of this paper. The research is supported by the National High Technology Research and Development Program of China (Grant No. 2012AA011201) and the National Natural Science Foundation of China (Grant No. 61432020 and 61472430).

References

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

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

[3] Dabbish L, Stuart C, Tsay J, and Herbsleb J, "Social coding in GitHub: transparency and collaboration in an open software repository," in *Proc. of the ACM 2012 conference on Computer Supported Cooperative Work*, 2012, pp. 1277-1286.

[4] Meeder B, Tam J, Kelley P G, and Cranor L F, "RT@ IWantPrivacy: Widespread violation of privacy settings in the Twitter social network," in *Proc. of the Web*, vol. 2, 2010.

[5] Louridas P, "Using wikis in software development," *Software*, vol. 23, no. 2, pp. 88-91, 2006.

[6] Park S and Maurer F, "The role of blogging in generating a software product vision," in *Proc. of the ICSE'09 Workshop on Cooperative and Human Aspects on Software Engineering*, 2009, pp. 74-77.

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

[8] Storey M A, Treude C, van Deursen A, and Cheng L T, "The impact of social media on software engineering practices and tools," in *Proc. of the FSE/SDP workshop on Future of software engineering research*, 2010, pp. 359-364.

[9] Begel A, DeLine R, and Zimmermann T, "Social media for software engineering," in *Proc. of the FSE/SDP workshop on Future of software engineering research*, 2010, pp. 33-38.

[10] Black S, Harrison R, and Baldwin M, "A survey of social media use in software systems development," in *Proc. of the 1st Workshop on Web 2.0 for Software Engineering*, 2010, pp. 1-5.

[11] Vega, Edgardo, Ramanujam Parthasarathy, and Josette Torres. "Where are my tweeps?: Twitter usage at conferences," *Paper, Personal Information*, pp. 1-6, 2010.

[12] Storey M, Ryall J, Singer J, Myers D, Cheng L T, and Muller M, "How software developers use tagging to support reminding and refinding," *IEEE Transactions on Software Engineering*, vol. 35, no. 4, pp. 470-483, 2009.

[13] Froehlich J and Dourish P, "Unifying artifacts and activities in a visual tool for distributed software development teams," in *Proc. of the 26th International Conference on Software Engineering (ICSE)*, 2004, pp. 387-396.

[14] Omoronyia I, Ferguson J, Roper M, and Wood M, "Using developer activity data to enhance awareness during collaborative software development," *Computer Supported Cooperative Work (CSCW)*, vol. 18, no. 5, pp. 509-558, 2009.

[15] Ahmadi N, Jazayeri M, Lelli F, and Nesic S, "A survey of social software engineering," in *Proc. of the Automated Software Engineering Workshops*, 2008, pp. 1-12.

[16] O'reilly T, "What is Web 2.0: Design patterns and business models for the next generation of software," *Communications and Strategies*, vol. 65, no. 1, pp. 17-37, 2007.

[17] Kotlarsky J and Oshri I, "Social ties, knowledge sharing and successful collaboration in globally distributed system development projects," *European Journal of Information Systems*, vol. 14, no. 1, pp. 37-48, 2005.

[18] Yang Zhang, Gang Yin, Yue Yu and Huaimin Wang, "Investigating Social Media in GitHub's Pull-requests: A Case Study on Ruby on Rails," in *Proc. of the 1th FSE'14 Workshop on Crowd Soft(CrowdSoft)*, 2014, Accepted.

[19] Gousios G and Spinellis D, "GHTorrent: Github's data from a firehose," in *Proc. of the 9th Working Conference on Mining Software Repositories (MSR)*, 2012, pp. 12-21.

[20] Gousios G, Vasilescu B, Serebrenik A, and Zaidman A, "Lean GHTorrent: GitHub data on demand," in *Proc. of the 11th Working Conference on Mining Software Repositories (MSR)*, 2014, pp. 384-38.