SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

Social media in GitHub: the role of @-mention in assisting software development

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*

Received January 1, 2016; accepted January 1, 2016; published online January 1, 2016

Abstract Recently, many studies have proposed the use of social media tools to promote the collaboration among developers, which is beneficial to software development. Nevertheless, there is insufficient empirical evidence to confirm a beneficial impact of using @-mention on the issues in GitHub. Therefore, we examined data from two large and successful projects hosted on GitHub: the Ruby on Rails and the AngularJS. By using qualitative and quantitative analyses, we provide an in-depth understanding on the use of @-mention to solve these issues and its role in assisting software development. Our statistical results indicate that @-mention attracts more participants and tends to be used for complex issues. In addition, @-mention favors solving issues by enlarging their visibility and facilitating the developers' collaboration. Further, we built an *@-network* based on the @-mention database we extracted. Moreover, we investigated the evolution of *@-network* over time and proved that we have the potential to mine the relationships and characteristics among developers by exploiting the knowledge from the *@-network*.

Keywords Issues, Social media, @-mention, GitHub, Software development

Citation Zhang Y, Wang H M, Yin G, et al. Social media in GitHub: the role of @-mention in assisting software development. Sci China Inf Sci, 2016, 59(1): xxxxxx, doi: xxxxxxxxxxxxxx

1 Introduction

Open-source software (OSS) enables anyone to be a part of the development process [1]. Owing to the decentralized, self-directed nature, and the social diversity in OSS, the success of an OSS project, to a large extent, depends on the social aspects of distributed collaboration and achieving coordination over distance [2]. To better support the collaboration and coordination, today's generation of developers frequently uses social media in their development environments [3]. The design of social media tools supports and promotes collaboration, often as a side-effect of individual activities, and democratizes who participates in activities previously in the control of just a few stakeholders [4]. These social media tools facilitate in simultaneously leveraging articulated social networks and observed code-related activity, which supports the awareness that is only available to core developers in previous [5]. Social media has changed the way people collaborate and share information on software development [6].

* Corresponding author (email: yangzhang15@nudt.edu.cn)

GitHub[1], a social collaborative software development community, has emerged and gained popularity in recent years. The platform integrates many social media tools to facilitate distributed collaboration, involving watch [7], follow [7], comment action [5], and @-mention [8]. @-mention is a typical social media used in the online social platforms such as Facebook[2], Twitter[3] and WeChat[4]. It allows users to reference a specific user by simply placing an "@" symbol before the username they wish to reference [9]. Compared to watch, follow, and other general social media like wikis [10], blogs [11] and microblogs [12], @-mention is usually generated from the description body or comments in the issue reports, which makes it more deeply involved in solving issues. Some studies showed that @-mention is a strong predictor of information diffusion [13] and is a significant factor in enlarging the visibility of a post and facilitating initiate responses and conversations [14]. Nevertheless, insufficient efforts have been made on analyzing the use of @-mention for considerable amount of issues, and whether their use has any impact on solving these issues in GitHub.

In this study, we examined data from two large and successful projects hosted on GitHub: Ruby on Rails and AngularJS. We use qualitative and quantitative approaches to conduct an in-depth exploration of @-mention in GitHub. Our results explicitly describe the current usage of @-mention and elicit some important implications for the developers to know better about the use of @-mention used in GitHub's issues. In particular, we studied (1) the use of @-mention in GitHub's issues (such as usage frequency, usage location, and usage scenarios), (2) the influence of @-mention on the solving of issues (such as the number of comments and time-to-solve) and (3) the understanding gained from the *@-network* (such as its evolution and mining influential developers).

Our results show that a significant difference exists in terms of issue characteristics between issues with and without @-mention. Usually, @-mention is used in the complex issues, and if present in a description body, it has more impact on the time-to-solve of issues than when present in comments. We prove that @-mention has a positive impact on the solving of issues by enlarging their visibility and facilitating developers' collaboration. Based on the *@-network* we built, we determined that the set of active @-mentioning node pairs change rapidly over time but the *@-network* basically retains stable structural properties. In addition, we ran the PageRank algorithm in the *@-network* to identify influential developers. The result proves that we can use *@-network* to determine the influential developers in GitHub's issues. Moreover, our study can help the developers and researchers notice the significance of @-mention in GitHub and make better use of it. Based on the *@-network*, we also propose an interesting and promising research direction to analyze the relationships and characteristics among developers.

The remainder of this paper is organized as follows. Section 2 presents the related work. In Section 3, we introduce related concepts and our research questions. Section 4 introduces our exploration and the main results. Section 5 discusses the threats to validity. Finally, we provide conclusions in Section 6.

2 Related work

To enhance the collaboration in software development, some studies proposed tagging [15], searchable graphs of heuristically linked artifacts [16], and workspace awareness [17] to support coordination. Storey *et al.* [4] investigated the benefits, risks, and limitations of using social media in software development at the team, project, and community levels. Kotlarsky *et al.* [18] proposed that social ties and knowledge contribute to successful collaboration in globally distributed information system development teams. In their study, they pointed that human-related issues involving rapport and transactive memory are important for collaborations in software development. Black *et al.* [19] described the preliminary results of a pilot survey conducted to collect information on the use of social media in the development of global software systems and determine that social media can enable better communication through the

1) https://github.com/
2) https://www.facebook.com/
3) https://www.twitter.com/
4) http://weixin.qq.com/

development of software systems. In particular, their research results showed that 91% of respondents claimed that the social media improved their working life.

O'Reilly [20] mentioned that social media tools can be characterized by an underlying "architecture of participation" that supports crowdsourcing and a many-to-many broadcast mechanism. Ahmadi *et al.* [3] determined that today's generation of developers frequently use social media to augment tools in their development environments. Park *et al.* [11] proved that blogs are frequently used by developers to document "how-to" information to discuss the release of new features and support requirement engineering. Louridas *et al.* [10] proposed that wikis are used to support defect tracking, documentation, requirements tracking, test-case management, and project portal creation. Riemer *et al.* [12] argued that decision makers should entrust their employees in putting microblogging to productive use in their group work environments. All these studies mainly focused on the correlation between the general social media and overall software development. Our study focuses on analyzing the influence of a special @-mention tool on the project issues 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 before the username [9]. Yang *et al.* [13] determined that @-mention is a strong predictor of information diffusion. Lumbreras *et al.* [21] proposed that @-mention usually expresses some type of close or familiar relationship and can be treated as a positive indicator of mutual trust. Further, Vega *et al.* [14] reported that @-mention is a significant factor in enlarging the visibility of a post and facilitating in initiating responses and conversations. In our previous study, we conducted a preliminary investigation of @-mention used in the pull-requests hosted in Ruby on Rails [8]. We extended this prior study and conducted an exploratory study of @-mention in pull-requests based software development, including its current situation and benefits [22]. We have proved that @-mention is beneficial for processing pull-requests in GitHub. However, general issues and pull-requests in GitHub are managed by an issue tracking system. It would be interesting to expand these studies by considering the @-mention used for total issues. Unlike our previous study, in this study, we comprehensively analyze @-mention in both general issues and pull-requests, and investigate the @-network, involving its network structure, evolution, and potential to mine the relationships and characteristics among developers during software development.

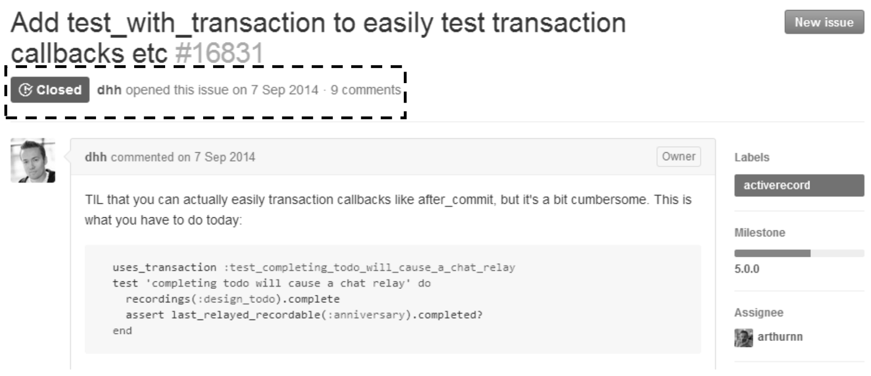
3 Background & Experimental Setup

In this section, we briefly introduce the issues and @-mention in GitHub. Next, we present our research questions and data set.

3.1 Issues

Reporting issues (bugs, new features or requirements) may be the most common contributions to problems in software development in different methods (*e.g.* testing, coding) [23]. These issues are managed by the issue tracking system, by providing a feature-rich interface. Generally, an issue consists of a title, description body, comments, and other additional information.

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 [24]. GitHub provides a light-weight and flexible issue tracking system, which provides the usual facilities for tracking issues, including filing and labeling issue tickets, setting a milestone, and submitting the comments. In our study, we divide the issues into *general-issues* and *pull-requests*. For example, 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*, as shown in the dashed boxes: compared to the *general-issues*, *pull-requests* have more information of commits in their interfaces. *Pull-requests* as implemented by GitHub in particular, is a new model for collaborating on distributed software development [25]. Moreover, it is maintained by the issue tracking system in GitHub. For each opened *pull-request*, an issue is opened automatically. Thus, every *pull-request* is an issue but not every issue is a *pull-request*. We can consider



Add test_with_transaction to easily test transaction callbacks etc #16831

New issue

Closed dhh opened this issue on 7 Sep 2014 · 9 comments

dhh commented on 7 Sep 2014 Owner

TIL that you can actually easily transaction callbacks like after_commit, but it's a bit cumbersome. This is what you have to do today:

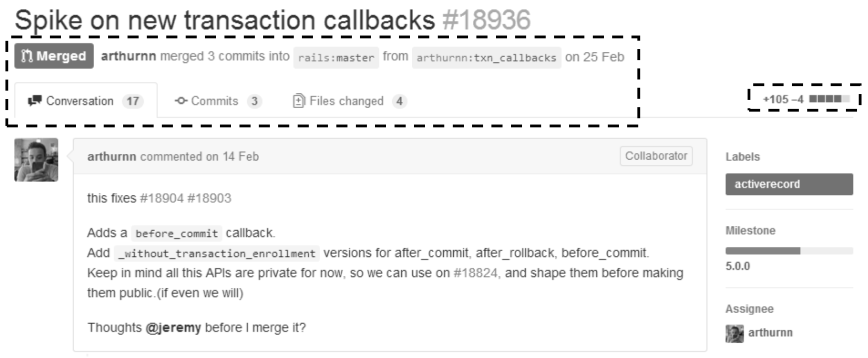
```
uses_transaction :test_completing_todo_will_cause_a_chat_relay
test 'completing todo will cause a chat relay' do
  recordings(:design_todo).complete
  assert last_relayed_recordable(:anniversary).completed?
end
```

Labels: activerecord

Milestone: 5.0.0

Assignee: arthurnn

(a) *general-issue*



Spike on new transaction callbacks #18936

Merged arthurnn merged 3 commits into `rails:master` from `arthurnn:txn_callbacks` on 25 Feb

Conversation 17 Commits 3 Files changed 4 +105 −4

arthurnn commented on 14 Feb Collaborator

this fixes #18904 #18903

Adds a `before_commit` callback.
Add `_without_transaction_enrollment` versions for after_commit, after_rollback, before_commit.
Keep in mind all this APIs are private for now, so we can use on #18824, and shape them before making them public.(if even we will)

Thoughts **@jeremy** before I merge it?

Labels: activerecord

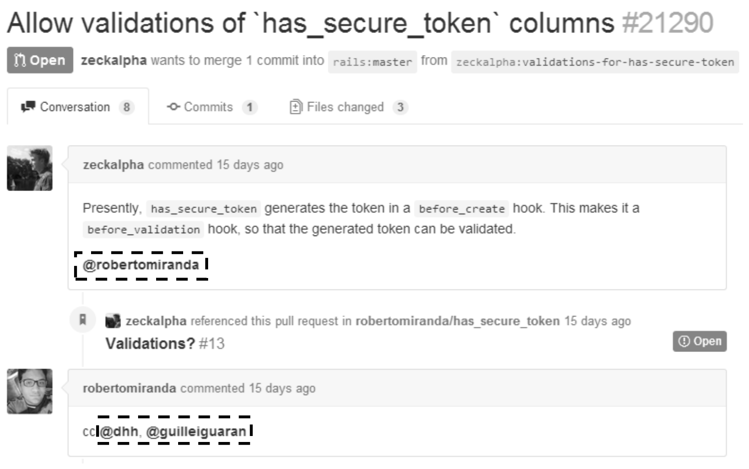
Milestone: 5.0.0

Assignee: arthurnn

(b) *pull-request*

Figure 1 Two examples of issues in GitHub.

the *pull-requests* as special issues in addition to the *general-issues*. Unlike the *general-issues*, *pull-requests* have two types of comments in addition to the *general comments* (basic type of comments on *general-issues* or *pull-requests*). These include (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*).



Allow validations of `has_secure_token` columns #21290

Open zeckalpha wants to merge 1 commit into `rails:master` from `zeckalpha:validations-for-has-secure-token`

Conversation 8 Commits 1 Files changed 3

zeckalpha commented 15 days ago

Presently, `has_secure_token` generates the token in a `before_create` hook. This makes it a `before_validation` hook, so that the generated token can be validated.

@robertomiranda

zeckalpha referenced this pull request in robertomiranda/has_secure_token 15 days ago

Validations? #13 Open

robertomiranda commented 15 days ago

cc @dhh, @guilleiguaran

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

3.2 @-mention

@[5], normally read as "at", especially in email addresses, implies "located at" or "directed at". Recently, an increasing number of online social platforms (*e.g.* Facebook, Twitter, and WeChat) use "@" to denote a reference or reply called as @-mention. This feature enables users to directly reference others by placing an "@" symbol before their username such as "@Jack". @-mention then automatically interprets these as links to the user's profile. In addition to the link function, after @-mentioning someone, the @-mentioned person could receive a reminder to respond immediately. In the issues of GitHub, @-mention can be used in the description body or comments of issues. As the "@" symbol in the title of issues is just a text and does not have any link function, we did not consider it in our study. Figure 2 shows an example of using @-mention in GitHub. In the figure, when *zeckalpha* reports this issue *#21290*, he refers @ *robertomiranda* in the issue's description body for reviewing. After *robertomiranda* reviews the new issue, he tags @ *dhh* and *guilleiguaran* for advice in this issue's comment.

3.3 Research Questions

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

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

To answer this question, we chose two famous and large projects hosted on GitHub for our investigation. We analyzed the usage of @-mention in the issues of the two projects, including the usage frequency, location, scenarios, and participants.

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

Here, we mainly focused on comparing the differences between issues with and without @-mention according to the following issue characteristics: the number of comments, number of participants, description complexity, and time-to-solve. We then used statistical tests to verify the significance of these differences.

RQ3: **@-network**. What can we learn from the *@-network*? Can we investigate the relationships and characteristics of developers by mining the *@-network*?

For this, we analyzed the basic network structural properties of *@-network* and generated many snapshots to capture the evolution of *@-network*: update frequency and network structure change. We then ran the PageRank algorithm in the *@-network* to identify the influential developers.

3.4 Research Data Set

We performed our empirical study on two famous projects: Ruby on Rails[6] (Rails) and AngularJS[7]. These are the largest and most successful projects hosted on GitHub. Rails is a web-application framework that includes everything needed to create database-backed web applications, according to the model-view-controller (MVC) pattern. In addition, it is one of the greatest open source projects used by GitHub to power its infrastructure. AngularJS is a toolset for building a framework most suited to web application development. Table 1 shows the basic information of Rails and AngularJS up to August 2015.

Table 1 Basic information of Rails and AngularJS.

Project	Language	Created_at	Watch	Star	Fork
Rails	Ruby	2008-04-11	2010	27534	11055
AngularJS	JavaScript	2010-01-06	4101	45188	20802

Our research database relies on the GitHub Rest API[8], which we can use to retrieve information on publicly accessible contents of the repositories of Rails and AngularJS. Up to August 2015, we collected

5) http://en.wikipedia.org/wiki/@
6) https://github.com/rails/rails
7) https://github.com/angular/angular.js
8) http://developer.github.com/

21,273 and 12,647 issues from Rails and AngularJS, respectively. The specific information of our research database is shown in Table 2, in which "#" implies "the number of ".

Table 2 The specific information of our research database.

Project	#issues	#comments	#developers
Rails	Total: 21273 *general-issues*: 7502, *pull-requests*: 13771	Total: 122935 *general comments*: 93711, *commit comments*: 12495, *pull-request review comments*: 16729	10094
AngularJS	Total: 12647 *general-issues*: 6679, *pull-requests*: 5968	Total: 62184 *general comments*: 54895, *commit comments*: 1286, *pull-request review comments*: 6003	9734

To analyze @-mention, we must extract its corresponding information (*e.g.* the @-mentioned developer and location) from the textual data of issues (description body and comments). As shown in Algorithm 1, the extraction can be divided into four steps: (1) For each project issue, we extract its description body and *general comments* information to build the basic textual data. We define these textual data as the *issueText*. (2) If this issue belongs to the *pull-requests*, we extract the *pull-request review comments* and *commit comments* and add them into the *issueText*, otherwise we directly go to the third step. (3) 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, that is, the username in the string "@username", otherwise we scan the next issue. (4) We then query the *Project Users* table to check whether the "@" is a valid @-mention action because some texts proceeding "@" are not real usernames, such as "Hongli Lai <hongli@phusion.nl>", which is an email address. If it is a valid @-mention action, we insert the valid @-mention information into our MySQL database for manipulation in the subsequent phases, otherwise we scan the next issue.

Algorithm 1 Extracting @-mention method

```
1:  Input: $Issues$ //$general$-$issues$ or $pull$-$requests$
2:  $@$-$mention\ database \leftarrow \emptyset$
3:  foreach $issue\ I_a$ in $Issues$ do:
4:      $descriptionText \leftarrow$ extractDescription($I_a$)
5:      $generalCommentsText \leftarrow$ extractGeneralComments($I_a$)
6:      $issueText \leftarrow descriptionText \cap generalCommentsText$
7:      if $I_a \in pull$-$requests$:
8:          $pullRequestReviewCommentsText \leftarrow$ extractPullRequestReviewComments($I_a$)
9:          $commitCommentsText \leftarrow$ extractCommitComments($I_a$)
10:         $issueText \leftarrow issueText \cap pullRequestReviewCommentsText \cap commitCommentsText$
11:     if $@ \in issueText$:
12:         $atDeveloperData \leftarrow$ extractAtDeveloperData($issueText$)
13:         foreach $developer\ D_a$ in $atDeveloperData$ do:
14:             if $D_a \in Project\ Users$:
15:                 $@$-$mention\ database \leftarrow \{I_a, D_a\}$
16: Output: $@$-$mention\ database$
```

4 Exploration of @-mention in GitHub

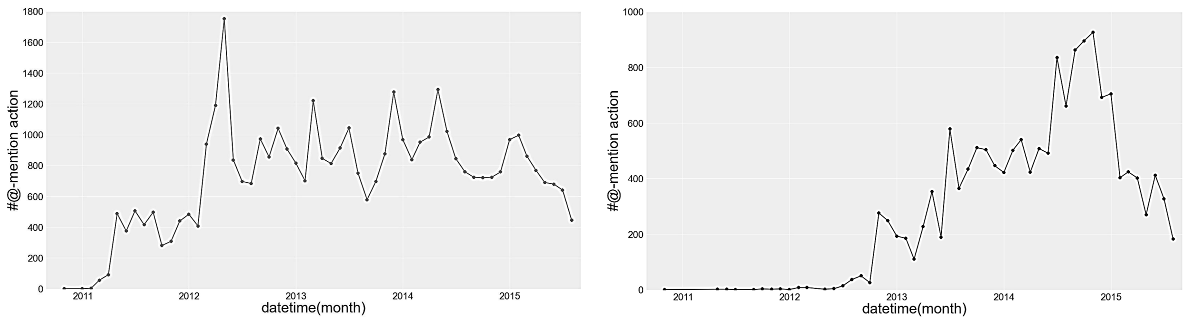
In this section, we present the results of our empirical study. These results are reported as responses to the research questions provided in Section 3.3.

4.1 RQ1: @-mention Usage

First, we investigated the usage of @-mention. In our study, we separately analyzed the usage frequency, location, scenarios, and participants in Rails and AngularJS.

4.1.1 *Usage frequency*

After parsing the *issueText* by using Algorithm 1, we extracted 41,395 and 15,668 @-mention actions in the Rails and AngularJS projects, respectively. We determined that 11,124 issues (52% of 21,273 issues) contain @-mention in Rails and 5332 issues (42% of 12,647 issues) contain @-mention in AngularJS. Figure 3 shows the number of @-mention actions that occur every month in Rails and AngularJS. On a monthly average, 726 and 307 @-mention actions occur in Rails and AngularJS, respectively. This result reveals that **@-mention is used quite frequently in large and famous projects**, which provides sufficient data for our investigation.



(a) Rails (b) AngularJS

Figure 3 Number of @-mention actions occurring monthly in Rails and AngularJS.

Actually, each issue may contain many @-mention actions. Therefore, we determined the average number of @-mention actions used in each issue. Table 3 shows the main statistical results. The first row (#@-mention) shows the number of @-mention actions used in each issue. The second and third rows show the specific number of issues (ratio) in Rails and AngularJS, respectively. As shown, the vast majority of issues (Rails: nearly 65%, AngularJS: nearly 75%) only have 1-3 @-mention actions, implying that **the frequency of each issue using @-mention is low**.

Table 3 Number of @-mentions used in issues.

#@-mention	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**>10**	**Total**
Rails	3612 (32.47%)	2245 (20.18%)	1399 (12.58%)	1016 (9.13%)	661 (5.94%)	475 (4.27%)	397 (3.57%)	292 (2.62%)	222 (2.00%)	161 (1.45%)	644 (5.79%)	11124
AngularJS	2283 (42.82%)	1084 (20.33%)	648 (12.15%)	377 (7.07%)	253 (4.74%)	162 (3.04%)	136 (2.55%)	92 (1.73%)	59 (1.11%)	45 (0.84%)	193 (3.62%)	5332

4.1.2 *Usage location*

As mentioned in Sections 3.1 and 3.2, @-mention is usually used in an issue's description body or in *general comments*. In *pull-requests*, @-mention is also used in *pull-request review comments* or *commit comments*. Therefore, we analyzed the specific location of @-mention used in issues. Figure 4 shows that, in Rails, 94% @-mention actions are used in the comments (88% in *general comments*) and only 6% @-mention actions are used in the description body. In AngularJS, 98% @-mention actions are used in the comments (95% in *general comments*) and only 2% @-mention actions are used in the description body. This indicates that **most @-mention actions are used in an issue's comments (especially in the *general comments*) rather than in an issue's description body, that is, @-mention is more likely to be used in developer conversations**.

4.1.3 *Usage scenarios*

In our study, we divide the scenarios of @-mention used in issues into two: "@ submitter" and "@ reviewer". "@ submitter" implies that the reviewer participated in the issue's discussion @-mention

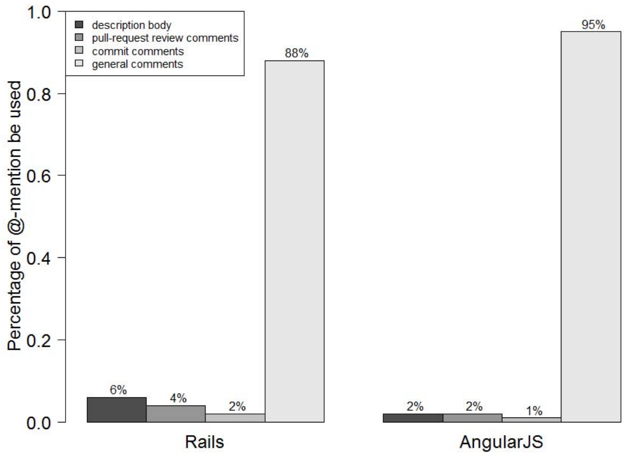


Figure 4 Distribution of specific location of @-mention used in issues.

an issue's submitter and "@ reviewer" implies that the issue's submitter @-mention the reviewer or the reviewers @-mention each other. We determined that 24% @-mention are used for "@ submitter", whereas 76% are used for "@ reviewer" in Rails. Further, in AngularJS, 31% @-mention are used for "@ submitter" whereas 69% are used for "@ reviewer". This indicates that **most @-mention actions are used by the issue's submitters to @-mention other developers for a review or by reviewers to @-mention each other for discussion**.

4.1.4 *Usage participants*

We determined that among the total 10,094 developers in Rails, 10,057 are not internal project members but contribute their codes and suggestions to the 37 internal project members. In AngularJS, 9699 external developers contribute to the 35 internal project members. The internal project members can access the repository of project and manage all the issues through internal or external developers.

Furthermore, we investigated the participants of the @-mention action. We divided the participants of @-mention used in issues into "@-mentioning developers" and "@-mentioned developers". The "@-mentioning developers" are the developers who @-mention other developers and "@-mentioned developers" are the developers who are @-mentioned by others. In our study, we determined that in Rails, 43% "@-mentioning developers" and 54% "@-mentioned" developers are internal project members. Each internal project member on an average can @-mention other developers 481 times and be @-mentioned 606 times, whereas each external developer on an average can only @-mention other developers 3 times and be @-mentioned twice. In AngularJS, 55% "@-mentioning developers" and 51% "@-mentioned developers" are internal project members. Thus, each internal project member on an average can @-mention other developers 246 times and be @-mentioned 228 times, whereas each external developer on an average can only @-mention other developers once and be @-mentioned once. This proves that **@-mention is generally used in the group of internal project members**.

Discussion:
As the results indicate, @-mention is used quite often in Rails and AngularJS probably because both are famous and large-scale open source projects that attract thousands of developers (internal or external) for contributions and online discussions. In particular, the mechanism of *pull-requests* leads to the internal project members requiring much resource and time to decide whether *pull-requests* should be merged into the core repository. As the issues require more discussions, @-mention is very likely to be used to involve more developers in the solving process. The specific location and scenario of @-mention reveals that @-mention is more likely to be used in the comments during the developers' conversation instead of in the issue's description body. Further, most @-mention actions are used by issue's submitter to @-mention other reviewers or reviewers to @-mention each other. We consider that it is difficult

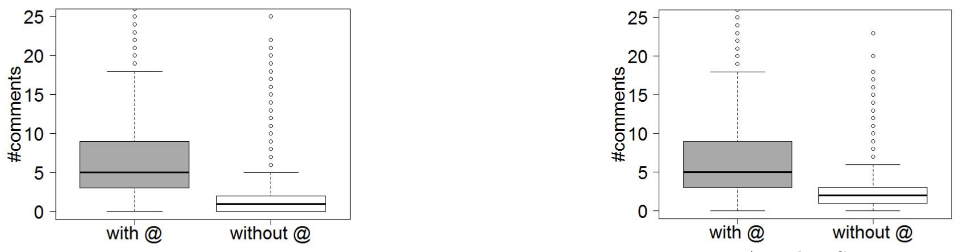
for an issue's submitter to @-mention suitable developers at the beginning of the issue's solving process. During the comment-based conversation, with the assistance of other participants, the @-mention problem could be solved easily. As GitHub provides a distributed collaborative software development pattern, all developers require their codes and suggestions to be accepted by the internal project members. Thus, all the development activities are performed around the internal project members. And that is why we find that @-mention is generally used in the group of internal project members.

4.2 RQ2: @-mention Influence

Based on the investigation of the use of @-mention in issues, we attempted to analyze if @-mention has a positive impact on the solving process of issues. We used the *R* statistical analysis tool to determine the differences between issues with and without @-mention. In addition, we used statistical tests including the Mann-Whitney-Wilcoxon (MWW) test, Z test, and Cliff's δ to validate the significance of these differences. All of these statistical tests are nonparametric statistical hypothesis tests. They do not assume any specific distribution, which is a suitable property for our experimental analysis.

4.2.1 *The number of comments*

First, we analyzed the distribution of the number of comments in issues with and without @-mention. Figure 5-a shows that in Rails, the average number of comments is 1.8 (median: 1.0) for issues without @-mention, whereas the value increases to 7.1 (median: 5.0) for issues with @-mention. Figure 5-b shows that in AngularJS, the average number of comments is 2.3 (median: 2.0) for issues without @-mention, whereas the value increases to 7.7 (median: 5.0) for issues with @-mention.



(a) Rails (b) AngularJS

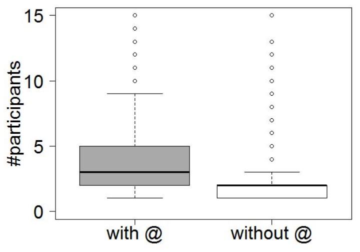
Figure 5 Number of comments in issues with and without @-mention[9].

By using statistical tests, we verify that the differences between issues with and without @-mention are statistically significant (Rails: W=94,755,000, p<2.2e-16, z=62.8 [very significant], δ=0.68 [large]; AngularJS: W=32,326,000, p<2.2e-16, z=36.0 [very significant], δ=0.66 [large]). This indicates that **issues with @-mention are likely to have more comments than issues without @-mention, that is, @-mention promotes the discussion of developers in issues**.

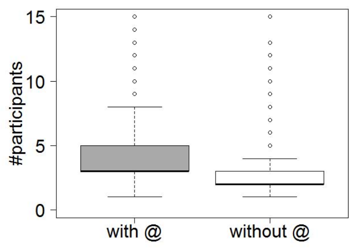
4.2.2 *The number of participants*

Next, we analyzed the distribution of the number of participants (submitter and reviewers) in issues with and without @-mention. Figure 6-a shows that in Rails, the average number of participants is 1.9 (median: 2.0) for issues without @-mention, whereas the value increases to 4.0 (median: 3.0) for issues with @-mention. Figure 6-b shows that in AngularJS, the average number of participants is 2.3 (median: 2.0) for issues without @-mention, whereas the value increases to 4.2 (median: 3.0) for issues with @-mention.

9) In the boxplot, there are five main horizontal lines. The top line indicates the maximum value. The second line indicates the upper quartile. The third line indicates the median value. The fourth line indicates the low quartile and the bottom line indicates the minimum value. All data points above the top line or below the bottom line are outliers (determined by the tool).



(a) Rails



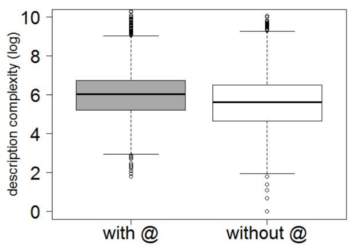
(b) AngularJS

Figure 6 Number of participants in issues with and without @-mention.

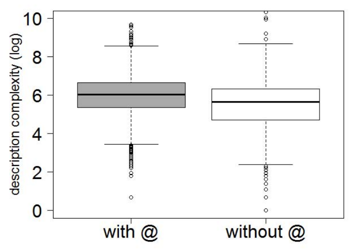
We tested and confirmed that the distributions between issues with and without @-mention are significantly different by using statistical tests (Rails: W=93,673,000, p<2.2e-16, z=71.5 [very significant], δ=0.66 [large]; AngularJS: W=31,022,000, p<2.2e-16, z=41.7 [very significant], δ=0.59 [large]). This indicates that **issues with @-mention are likely to have more participants than issues without @-mention, that is, @-mention facilitates developers to participate in the solving of issues**.

4.2.3 *The description complexity*

Next, we analyzed the description complexity (the textual length of title and issue's description body) of issues with and without @-mention. Figure 7-a shows that in Rails, the average description complexity is 618.3 (median: 276.0) for issues without @-mention, whereas the value increases to 772.8 (median: 421.0) for issues with @-mention. Figure 7-b shows that in AngularJS, the average description complexity is 455.8 (median: 283.0) for issues without @-mention, whereas the value increases to 639.0 (median: 424.0) for issues with @-mention.



(a) Rails



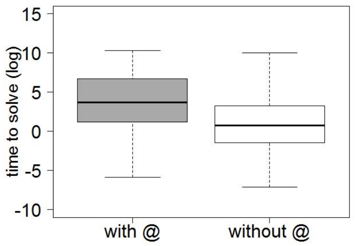
(b) AngularJS

Figure 7 Description complexity in issues with and without @-mention.

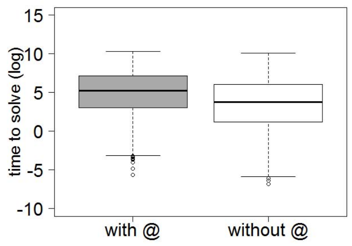
The statistical tests show that the differences between issues and without @-mention are statistically significant except for the value of cliff's δ (Rails: W=66,571,000, p<2.2e-16, z=7.2 [very significant], δ=0.18 [small]; AngularJS: W=23,824,000, p<2.2e-16, z=10.0 [very significant], δ=0.22 [small]). This indicates that **issues with @-mention may have more description text than issues without @-mention, that is, issues with considerable description text are more likely to use @-mention than issues with less description**.

4.2.4 *The time-to-solve*

We also studied the distribution of the time-to-solve (time interval between an issue is opened and closed) in issues with and without @-mention. In our study, we only focused on the closed issues. Figure 8-a shows that in Rails, the average time to solve the issues with @-mention is 1330.7 h (median: 40.6 h), whereas the time reduces to 312.0 h (median: 2.2 h) for issues without @-mention. Figure 8-b shows that in AngularJS, the average time to solve the issues with @-mention is 1463.5 h (median: 186.8 h), whereas the time reduces to 1143.1 h (median: 42.0 h) for issues without @-mention. The results of our statistical tests prove that these differences are statistically significant (Rails: W=73,391,000, p<2.2e-16, z=31.2 [very significant], δ=0.42 [medium]; AngularJS: W=19,115,000, p<2.2e-16, z=5.5 [very significant], δ=0.23 [small]). This indicates that **issues with @-mention are likely to need more time to be solved than issues without @-mention**.



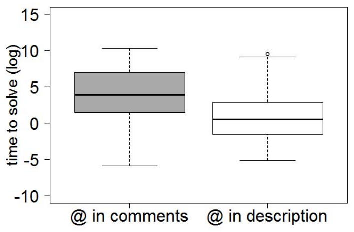
(a) Rails



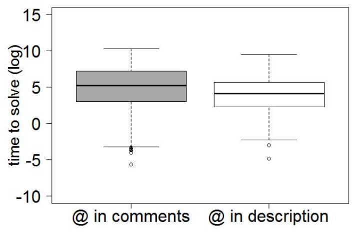
(b) AngularJS

Figure 8 Time-to-solve in issues with and without @-mention.

As mentioned earlier, @-mention can be used in an issue's description body or comments. Therefore we analyzed the impact of a specific location of @-mention on the time-to-solve. We filtered out two groups from the closed issues: issues with @-mention only in their description body and only exist in their comments. Figure 9-a shows that in Rails, the average time to solve the issues with @-mention only in the description body is 198.7 h (median: 1.7 h), whereas the time increases to 1462.6 h (median: 50.2 h) for issues with @-mention only in comments. Figure 9-b shows that in AngularJS, the average time to solve the issues with @-mention only in description body is 678.4 h (median: 62.3 h), whereas the time increases to 1498.2 h (median: 190.8 h) for issues with @-mention only in comments. The results of our statistical tests prove that these differences are statistically significant (Rails: W=4,527,300, p<2.2e-16, z=24.0 [very significant], δ=0.50 [large]; AngularJS: W=252,510, p<2.2e-16, z=3.7 [very significant], δ=0.23 [small]). This indicates that **@-mention in the description body has more impact on the solving of issues than @-mention in comments**.



(a) Rails



(b) AngularJS

Figure 9 Time-to-solve in issues with @-mention only in description body and only in comments.

4.2.5 *Other characteristics of issues*

We analyzed other characteristics (label, milestone, assignee, and state) of issues with and issues without @-mention. Table 4 shows that in Rails, 43.9% issues with @-mention have labels, whereas the ratio for issues without @-mention is only 14.2%. In AngularJS, 71.5% issues with @-mention have labels, whereas the ratio for issues without @-mention is only 53.0%. Similarly, issues with @-mention are more likely to have a milestone and assignee than issues without @-mention both in Rails and AngularJS. The results of Chi-squared tests show that the differences in proportion between the issues with and without @-mention are significant. These results reveal that **issues with @-mention are more likely to have labels, milestone as well as assignees than issues without @-mention**. As aforementioned, issues with @-mention need more time to be solved than issues without @-mention. Thus, in both Rails and AngularJS, the ratio of closed issues with @-mention is slightly lesser than issues without @-mention.

Table 4 Other characteristics of issues with and without @-mention.

Characteristics	Issues in Rails			Issues in AngularJS		
	With @-mention	**Without @-mention**	**Chi-Squared Test**	**With @-mention**	**Without @-mention**	**Chi-Squared Test**
have label	43.9%	14.2%	2248.80 ***	71.5%	53.0%	444.50 ***
have milestone	9.3%	3.0%	361.95 ***	54.6%	30.3%	752.52 ***
have assignee	8.8%	2.5%	389.60 ***	25.4%	14.0%	260.00 ***
issue:closed	94.5%	97.2%	97.34 ***	87.2%	91.6%	64.29 ***
'***' p<0.001, '**' p<0.01, '*' p<0.05, '.' p<0.1						

Discussion:

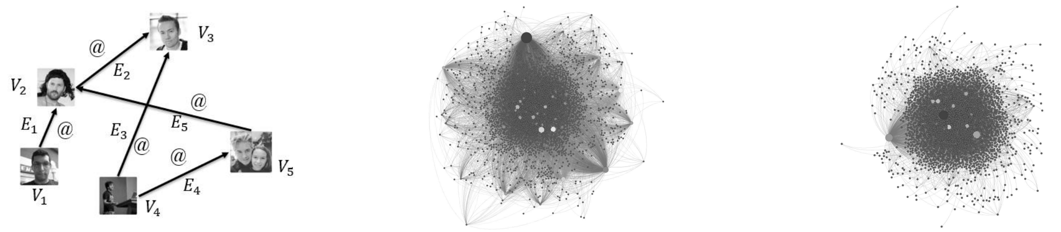
By using statistical tests, we determined that issues with @-mention may have more comments, more participants, more description texts, and a longer time-to-solve than issues without @-mention, indicating that @-mention is very likely to be used in complex issues. We assume that difficult issues require more time and resources to be solved. However, the @-mention mechanism can promote the discussion of developers in issues and facilitate developers to participate in their solving process. We further determined that @-mention used in an issue's description body greatly affects the time-to-solve of issues than @-mention in comments. This may be because @-mention in the description body can enlarge the visibility of issues, thus highlighting it to the developers. In addition, we determined that issues with @-mention are more likely to have a label, milestone, and assignee than issues without @-mention, implying that issues with @-mention get more attention and appropriate management than issues without @-mention. These results prove that @-mention helps involve more participants and positively affects the solving of issues.

4.3 RQ3: @-network

Our investigations show that @-mention has many possible research directions. In particular, @-mention builds a social network among the developers for solving issues, comprising rich information for mining and research.

4.3.1 *Building @-network*

We first constructed a social network based on our @-mention database, which we called *@-network*. The *@-network* can be defined as a directed graph $G_{fn} = (V, E)$, where V represents the set of vertices, representing all developers participating in the project, and 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, an edge is formed from v_i to v_j. For example, in Figure 10-a, V_2 is @-mentioned by V_1, thus, a directed edge E_1 is formed from V_1 to V_2. Figures 10-b and 10-c show the total *@-network* redrawn by *Force Atlas*[10] algorithm in Rails and AngularJS, respectively.



(a) An example of @-network (b) @-network in Rails (c) @-network in AngularJS

Figure 10 Building @-network in Rails and AngularJS.

The *@-network* of Rails consists of 5059 nodes and 17706 edges and the *@-network* of AngularJS consists of 3833 nodes and 8053 edges. As shown in Table 5, we computed the basic network structural properties of *@-network* in Rails and AngularJS. The average *degree* computed in the *@-network* of Rails is 3.5 (*indegree*: 1.7, *outdegree*: 1.8), implying that each developer may @-mention other developers 1.8

10) https://github.com/gephi/gephi/wiki/Force-Atlas-2

Table 5 Basic structural properties of @-network in Rails and AngularJS.

Project	#nodes	#edges	Avg. degree	Avg. clustering coeff.	Avg. shortest path	Diameter
Rails	5059	17706	3.5	0.27	3.19	10
AngularJS	3833	8053	4.2	0.20	3.15	10

times and be @-mentioned 1.7 times. In the *@-network* of AngularJS, the average *degree* is 4.2 (*indegree*: 2.1, *outdegree*: 2.1). We followed the research direction of Surian *et al.* [26] and Leskovec *et al.* [27] and computed the shortest path between two nodes by ignoring the weights of the graph's edges. The length of a path between two nodes is simply the length of the series of nodes between them. In the *@-network* of Rails and AngularJS, the average shortest path is 3.19 and 3.15, respectively. Surian *et al.* [26] studied the Sourceforge[11] project hosting platform and determined that the average shortest path among project developers is 6.55, following the popular assumption of "six-degree-separation" [28]. Another study involving the Facebook social graph concluded that individuals on Facebook have a potentially tremendous reach with an average shortest path of 4.7 [29]. The average shortest path in *@-network* is significantly lower, suggesting that the **social media tool @-mention actually enables more collaboration among developers. Further, the *@-network* allows for even better reach as developers' relationships are tighter than human' relationships in daily life social networks.**

4.3.2 *@-network evolution over time*

We further analyzed the impact of the dynamics of the *@-network*. In particular, we are interested in capturing (a) how @-mentioning node pairs (v_i, v_j) in the *@-network* come and go and (b) to what extent overall properties of the *@-network* change over time.

We analyzed the evolution of the *@-network* over time by deriving multiple snapshots of the network. We generated these snapshots based on the @-mention action at 90-day intervals. Section 4.1.1 shows that only a small amount of @-mention actions occurred before May 2011 in Rails and November 2012 in AngularJS. Thus, in our study, we generated 17 snapshots of the *@-network* in Rails (from May 2011 to August 2015) and 11 snapshots in AngularJS (from November 2012 to August 2015). Each snapshot contains all the @-mentioning node pairs (v_i, v_j) during the 90 days before this snapshot. We examined the evolving *@-network* based on these snapshots.

(a) Update frequency of the @-network

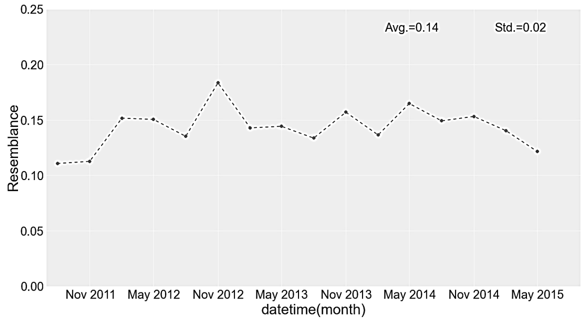
First, we investigated the update frequency of the *@-network* from one snapshot to the next. Similar to the approach proposed by Viswanath *et al.* [30], we used the notion of *resemblance* to measure the overlap in network links in two consecutive snapshots. In our study, *resemblance* is defined as the proportion of the @-mentioning node pairs that remain unchanged over two consecutive snapshots. The *resemblance* R_t at time t can be computed by the following equation, in which P_t represents the set of @-mentioning node pairs active at time t. The value of R_t varies from 0 to 1. $R_t = 1$ indicates the entire set of active @-mentioning node pairs that continued to interact at the next period, whereas $R_t = 0$ indicates that none of the @-mentioning node pairs that interacted in time t interacted in time $t + 1$.

$$R_t = |\frac{P_t \cap P_{t+1}}{P_t}|. \tag{1}$$

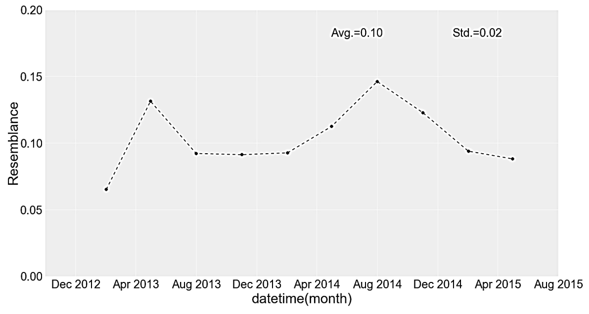
Figure 11-a shows that in Rails, the average *resemblance* across all snapshots is 0.14 (*Std.* = 0.02), indicating that on an average, only 14% of the @-mentioning node pairs remain active over time and 86% no longer exist in the consecutive snapshot. Figure 11-b shows that in AngularJS, the average *resemblance* is 0.10 (*Std.* = 0.02), indicating that only 10% of the @-mentioning node pairs remain active over time and 90% no longer exist in the consecutive snapshot. This reveals that **the @-mentioning node pairs in the *@-network* change dynamically over time.**

Furthermore, we mined the *companion* that existed in the *@-network*, where *companion* implies the set of @-mentioning node pairs that exist in a long period (two or three years). Table 6 shows that in Rails, only 11 (0.06%) *companions* exist in at least 12 snapshots (more than three years). In AngularJS,

11) http://sourceforge.net/



(a) Rails



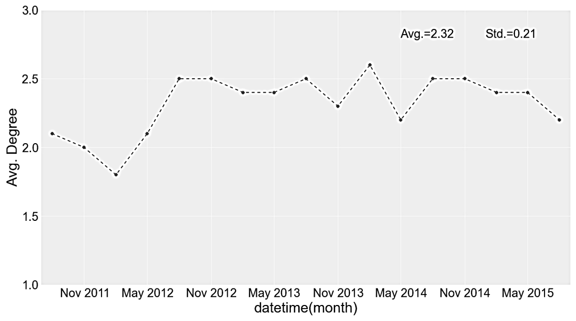
(b) AngularJS

Figure 11 Resemblance of the *@-network*.

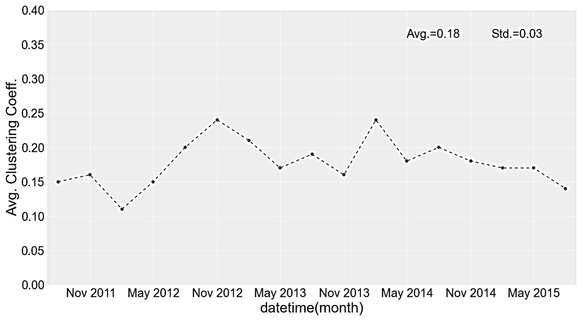
only 4 (0.05%) *companions* exist in at least 8 snapshots (more than two years). This result indicates that **only a small fraction of the *@-network* is stable in a long period.**

Table 6 Companion in Rails and AngularJS.

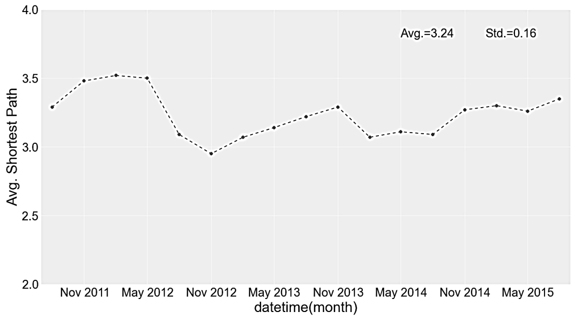
Project	Duration(month)	Time Period	Companion
Rails	45	2011-11-1 to 2015-8-1	(rafaelfranca, fxn),(rafaelfranca, tenderlove)
Rails	42	2012-2-1 to 2015-8-1	(rafaelfranca, jeremy),(rafaelfranca, pixeltrix),(rafaelfranca, senny)
Rails	39	2011-11-1 to 2015-2-1	(rafaelfranca, josevalim)
Rails	39	2012-5-1 to 2015-8-1	(rafaelfranca, dhh),(senny, rafaelfranca)
Rails	36	2011-5-1 to 2014-5-1	(guilleiguaran, tenderlove)
Rails	36	2012-8-1 to 2015-8-1	(senny, fxn),(senny, jeremy)
AngularJS	33	2012-11-1 to 2015-8-1	(petebacondarwin, IgorMinar)
AngularJS	27	2012-11-1 to 2015-2-1	(IgorMinar, petebacondarwin)
AngularJS	27	2013-5-1 to 2015-8-1	(btford, IgorMinar),(petebacondarwin, matsko)



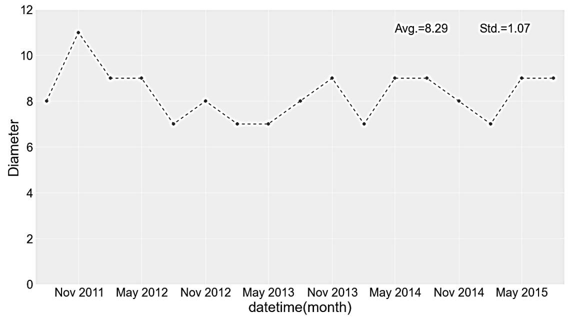
(a) *average degree*



(b) *average clustering coefficient*



(c) *average shortest path*



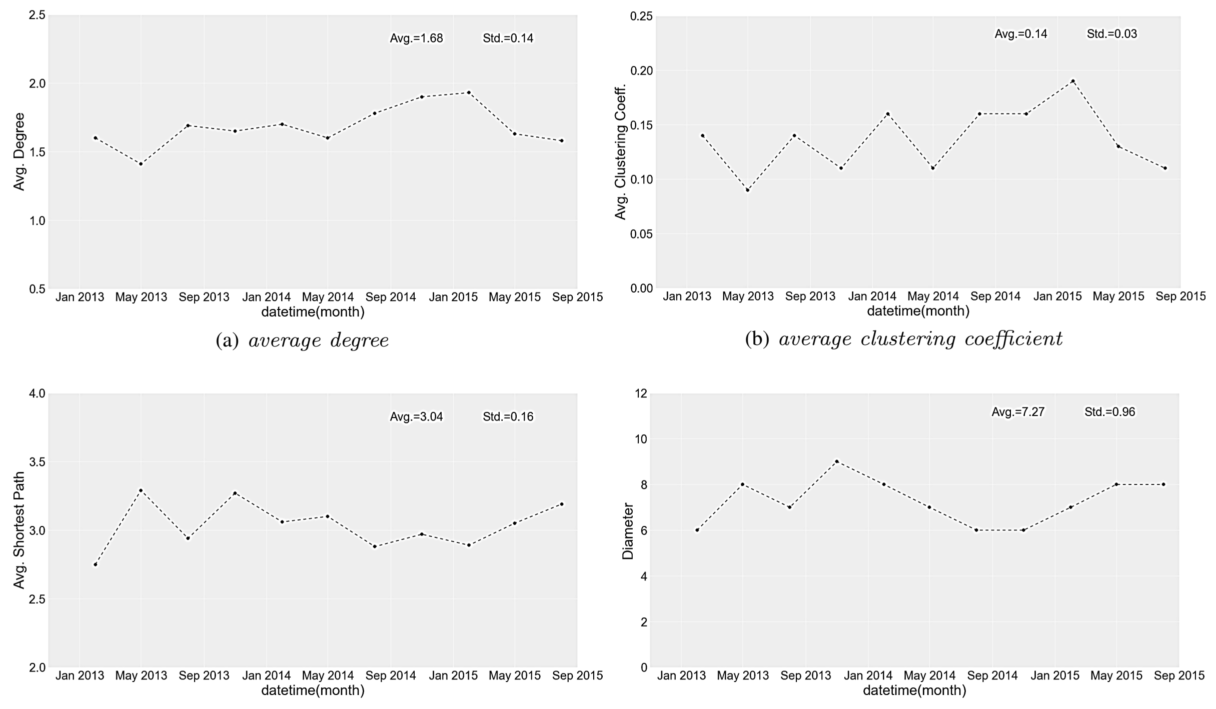
(d) *diameter*

Figure 12 Structural properties of the evolving *@-network* in Rails.

(b) Network structure change of the @-network

We further investigated to what extent network structural properties of the *@-network* change over time. To provide a detailed description, we calculated four popular network metrics in our study: average degree, clustering coefficient, average shortest path and diameter for each of our *@-network* snapshots.

Figure 12 shows that in Rails, the average degree is 2.32 (*Std.*=0.21), average clustering coefficient is 0.18 (*Std.*=0.03), average shortest path is 3.24 (*Std.*=0.16), and average diameter is 8.29 (*Std.*=1.07). In addition, Figure 13 shows in AngularJS, the average degree is 1.68 (*Std.*=0.14), average clustering coefficient is 0.14 (*Std.*=0.03), average shortest path is 3.04 (*Std.*=0.16), and average diameter is 7.27 (*Std.*=0.96). By analyzing that four network metrics vary over the snapshots in Rails and AngularJS, we determined that the different network measures are all stable over time with some small fluctuations. Based on our findings about the rapid update frequency of the *@-network*, our results on the network structural properties of the evolving *@-network* indicate that **the set of active @-mentioning node pairs change rapidly over time, but the *@-network* holds stable structural properties.**



(a) *average degree*

(b) *average clustering coefficient*

(c) *average shortest path*

(d) *diameter*

Figure 13 Structural properties of the evolving *@-network* in AngularJS.

4.3.3 *Identifying the influential developers*

We want to identify the influential developers from the *@-network*. Therefore, we used the PageRank algorithm in the *@-network*. PageRank algorithm, which is used for weighting the importance of web pages based on their links, has gained popularity because of its use in the Google search engine [31]. 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. The subsequent interactions then update these scores: the score of a developer d is distributed to the developers that d @-mention to. Each @-mentioned developer receives $\frac{1}{|L_d|}$ 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.

$$PR(d,i) = \frac{1-r}{T} + r \sum_{q \in K_d} \frac{PR(q,i-1)}{|L_q|}. \tag{2}$$

The PageRank algorithm returns a PageRank score for every developer. We can separately obtain the top-10 influential developers in Rails and AngularJS in terms of their PageRank scores. Table 7 and Table 8 show the results. The top-1 developer in Rails is "rafaelfranca", who is one of the internal project members of Rails. This developer submitted 2226 commits and 12020 comments, indicating that he is an active developer and has a lot of contribution to Rails. The top-1 developer in AngularJS is "caitp", who has submitted 263 issues and 6594 comments, revealing that he is an active and influential developer in AngularJS.

Table 7 Top-10 influential developers in Rails.

No.	Id	Developer	#commits	#comments	#general-issues	#pull-requests	#issues	Page_rank_value
1	7468	rafaelfranca	2226	12020	9	107	126	0.041331
2	8994	tenderlove	2106	1971	7	4	11	0.018902
3	8261	senny	923	5352	8	221	229	0.018666
4	1438	carlosantoniodasilva	696	4482	1	93	94	0.015479
5	7263	pixeltrix	223	2184	16	6	22	0.015224
6	8723	steveklabnik	88	2983	5	63	68	0.012908
7	4635	josevalim	1123	2874	7	6	13	0.012342
8	4289	jeremy	1152	1899	4	9	13	0.010521
9	8308	sgrif	481	2269	3	243	246	0.009154
10	3183	fxn	795	1998	4	2	6	0.008907

Table 8 Top-10 influential developers in AngularJS.

No.	Id	Developer	#commits	#comments	#general-issues	#pull-requests	#issues	Page_rank_value
1	790	caitp	123	6594	20	243	263	0.042881
2	3991	petebacondarwin	282	5067	25	133	158	0.031229
3	4054	pkozlowski-opensource	31	2654	6	53	59	0.023485
4	2170	IgorMinar	843	4481	146	248	394	0.023293
5	3245	matsko	202	1816	35	236	271	0.015898
6	3635	Narretz	45	2049	29	49	78	0.014093
7	755	btford	90	1968	12	56	68	0.013211
8	1893	gkalpak	28	1611	8	64	72	0.011763
9	2983	lgalfaso	44	1846	6	112	118	0.010537
10	3394	mhevery	434	742	104	89	193	0.006830

This result reveals that developers who are active or contribute more to the project obtain higher PageRank scores than developers who are not active and have less contribution. By analyzing the correlation between the PageRank scores and characteristics of developers, we determined that the characteristics of developers are consistent with the PageRank scores, that is, **we can evaluate the contribution, activeness, and influence of developers by mining the *@-network***.

Discussion:
@-mention builds a social network among the developers while solving issues, which contain rich information for mining. By analyzing this *@-network*, we determined that *@-network* allows for an even better reach as developers' relationships are stronger than human' relationships in daily life social networks. We believe that all the developers participating in the solving of issues are very determined. A strong purpose enables developers to interact more closely. Through the evolution of *@-network*, we analyzed the refreshing rate of @-mentioning node pairs and the resulting global network structural properties. This result reveals that while the individual @-mentioning node pairs constituting the *@-network* change over time, the average network structural properties remain relatively stable. Furthermore, in the *@-network*, the PageRank score of developers are basically consistent with their characteristics of software development. This proves that we can identify influential developers in software development. We certainly have the potential to assist software development by exploiting the knowledge from the *@-network*. Nevertheless, more research must be conducted to confirm and expand these results.

5 Threats to validity

In this section, we discuss the internal validity and external validity of our study.

5.1 Internal validity

Our statistical analysis mainly used factors such as the number of comments and, number of participants as measurements to verify the impact of @-mention on issues. Future studies must analyze some other characteristics of issues, for example, the number of files changed and code churn in *pull-requests*. As @-mention can be used in variably, for example, by expression of disagreement or notification. We cannot be sure, without further analysis, whether an @-mention action expresses trust, distrust, or none. We must conduct an in-depth analysis on the specific expression of @-mention in GitHub's issues.

5.2 External validity

We determined that some abnormal issues exist in simple problems but have a long handling time. In addition, some difficult issues are solved quickly because they are related to the upcoming release milestone of the project. Some issues represent new feature requirements and some are real bugs. The different types of issues may lead to bias in our study.

6 Conclusions

In this paper, we gained a deep understanding of the role of @-mention in assisting software development in the issues of GitHub. We examined data from two large and famous projects Rails and AngularJS, including @-mention usage, @-mention influence, and the characteristics of *@-network*. Through statistical analysis, we determined that @-mention, as a widely used social media tool in online social platform, is considerably used in the software development in GitHub. Most @-mention actions are used in issues' comments for "@ reviewer" and are basically used in the group of internal project members. Furthermore, @-mention is determined to be beneficial in involving more collaboration and tends to be used in the difficult issues. We proved that @-mention enlarges the visibility of issues and facilitates developers' discussion. Moreover, issues with @-mention are more likely to have a label, milestone, and assignee than issues without @-mention. Our investigation shows that @-mention has a positive impact on the issue solving. Based on the @-mention database, we built a *@-network* for mining the relationships and characteristics among developers. Our analysis indicates that although a high churn exists in the @-mentioning node pairs that interact over time, many of the global structural properties of *@-network* remain relatively constant. Our study proves that we certainly have the potential to assist in software development by exploiting the knowledge from the *@-network*. The future study will focus on expanding our investigation around the study of *@-network*.

Acknowledgements This work was supported by the National Natural Science Foundation of China (Grant No.61432020, 61472430 and 61502512) and the Postgraduate Innovation Fund (Grant No.CX2015B028).

Conflict of interest The authors declare that they have no conflict of interest.

References

1 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. IEEE, 2007. 6-6
2 Bogdan V. Human Aspects, Gamificaiton, and Social Media in Collaborative Software Engineering. In: Proceedings of the 36th International Conference on Software Engineering. ACM, 2014. 646-649
3 Ahmadi N, Jazayeri M, Lelli F, et al. A survey of social software engineering. In: Proceedings of Automated Software Engineering Workshops. IEEE, 2008. 1-12
4 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. ACM, 2010. 359-364

5 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. ACM, 2012. 1277-1286

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

7 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. IEEE, 2013. 65-72

8 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. ACM, 2014. 37-41

9 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. 2010. 2:1-2

10 Louridas P. Using wikis in software development. IEEE Trans Software, 2006, 23(2): 88-91

11 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. IEEE, 2009. 74-77

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

13 Yang J, Counts S. Predicting the Speed, Scale, and Range of Information Diffusion in Twitter. In: Proceedings of ICWSM. AAAI, 2010. 355-358

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

15 Storey M A, Ryall J, Singer J, et al. How software developers use tagging to support reminding and refinding. IEEE Trans Software Engineering, 2009, 35(4): 470-483

16 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. IEEE, 2004. 387-396

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

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

19 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. ACM, 2010. 1-5

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

21 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. IEEE, 2012. 1159-1164

22 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. IEEE, 2014. 343-350

23 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. IEEE, 2015. 550-554

24 Thung F, Bissyand 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. IEEE, 2013. 323-326

25 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. ACM, 2014.

26 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. IEEE, 2010. 269-273

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

28 Travers J, Milgram S. An experimental study of the small world problem. Sociometry, 1969, 425-443

29 Ugander J, Karrer B, Backstrom L, et al. The anatomy of the facebook social graph. arXiv preprint, 2011, 1111-4503

30 Viswanath B, Mislove A, Cha M, et al. On the evolution of user interaction in facebook. In: Proceedings of the 2nd ACM workshop on Online social networks. ACM, 2009. 37-42

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