SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

March 2017, Vol. 60 032102:1–032102:18
doi: 10.1007/s11432-015-1024-6

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

Yang ZHANG*, Huaimin WANG, Gang YIN, Tao WANG & Yue YU

Key Laboratory of Parallel and Distributed Computing, College of Computer, National University of Defense Technology, Changsha 410073*, China*

Received June 13, 2016; accepted September 17, 2016; published online December 9, 2016

Abstract Recently, many researches propose that social media tools can promote the collaboration among developers, which are beneficial to the software development. Nevertheless, there is little empirical evidence to confirm that using @-mention has indeed a beneficial impact on the issues in GitHub. In order to begin investigating such claim, we examine data from two large and successful projects hosted on GitHub, the Ruby on Rails and the AngularJS. By using qualitative and quantitative analysis, we give an in-depth understanding on how @-mention is used in the issues and the role of @-mention in assisting software development. Our statistical results indicate that, @-mention attracts more participants and tends to be used in the difficult issues. @-mention favors the solving process of issues by enlarging the visibility of issues and facilitating the developers' collaboration. Our study also build an @-network based on the @-mention database we extracted. Through the @-network, we investigate its evolution over time and prove that we certainly have the potential to mine the relationships and characteristics of 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, 2017, 60(3): 032102, doi: 10.1007/s11432-015-1024-6

1 Introduction

Open-source software (OSS) enables anyone to be part of the development process as large [1]. Because of the decentralised, self-directed nature and the social diversity in OSS, the success of an OSS project depends to a large extent on the social aspects of distributed collaboration and achieving coordination over distance [2]. In order to better support the collaboration and coordination, today's generation of developers frequently makes use of social media in their development environments [3]. Social media tools design supports and promotes collaboration, often as a side effect of individual activities, and furthermore democratizes who participates in activities that were previously in the control of just a few stakeholders [4]. These social media tools make it possible to leverage articulated social networks and observed code-related activity simultaneously, which supports the type of awareness that only available to core developers in previous [5]. Social media has changed the way that people collaborate and share information in the 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 in front of 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 usually comes from the description body or comments in the issue reports, which makes it more deeply involved in the solving process of issues. Some researches presented that @-mention is a strong predictor of information diffusion [13] and is a significant factor in enlarging the visibility of a post and helping initiate responses and conversations [14]. Nevertheless, little effort has been done on analyzing how @-mention is used in a population of issues and on whether their use has any impact on the solving process of issues in GitHub.

In this paper, we examine data from two large and successful projects hosted on GitHub, the Ruby on Rails and the AngularJS. We use qualitative and quantitative approaches to conduct an in-depth exploration of @-mention in GitHub. Our results give an explicit description of the current usage of @-mention and elicit some important implications for the developers to know better of @-mention used in the issues of GitHub. In particular, we have studied (1) how @-mention is used in GitHub's issues (usage frequency, usage location, usage scenarios, etc.), (2) how @-mention may influence the solving process of issues (the number of comments, the time to solve, etc.) and (3) what we can learn from the @-network (its evolution, mining the influential developers, etc.).

Our results show that there exists a significant difference in terms of the characteristics of issues between issues that do not have @-mention and that do. @-mention tends to be used in the difficult issues and @-mention in description body has more impact on the time-to-solve of issues than @-mention in comments. We prove that @-mention has a positive impact on the solving process of issues by enlarging the visibility of issues and facilitating the developers' collaboration. Based on the @-network we build, we find that the set of active @-mentioning node pairs change rapidly over time, but the @-network basically holds stable structural properties. We also run the PageRank algorithm in the @-network to identify the influential developers. The result proves that we can use @-network to find the influential developers in GitHub's issues. Our work helps 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 of developers.

This paper is organized as follows. In Section 2 we describe 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. We conclude the article in Section 6.

2 Related work

In order to enhance the collaboration in software development, some research proposed the tagging [15], searchable graphs of heuristically linked artifacts [16], and workspace awareness [17] to support the 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 made the point that human-related issues involving rapport and transactive memory were important for collaborative work in the software development. Black et al. [19] described the preliminary results of a pilot survey conducted to collect information on social media use in global software systems development and found that social media can enable better communication

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

through the software system development process. In particular, their research results showed that 91% of respondents said that the social media has improve their working life.

As mentioned from O'reilly [20], social media tools can be characterized by an underlying "architecture of participation" that supports crowdsourcing as well as a many-to-many broadcast mechanism. Ahmadi et al. [3] found that today's generation of developers frequently makes use of 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 to support requirements engineering. Louridas [10] proposed that wikis are used to support defect tracking, documentation, requirements tracking, test case management and for the creation of project portals. Riemer et al. [12] argued that decision makers should vest trust in their employees in putting microblogging to productive use in their group work environments. Basically, these researches mainly focused on the correlation between the general social media and the overall software development. Our work is focused on analyzing how the special @-mention tool influence the issues of projects in GitHub.

@-mention, a typical social media tool used in social networking websites, e.g., Facebook and Twitter, allows users to reference a specific user by simply placing an "@" symbol in front of the username they wish to reference [9]. Yang et al. [13] found that @-mention is a strong predictor of information diffusion. Lumbreras et al. [21] proposed that @-mention usually express some kind of close or familiar relationships and can be treated as a positive indicator of mutual trust. The study presented by Vega et al. [14] reported that @-mention is a significant factor in enlarging the visibility of a post and helping initiate response and conversations. In our previous work, we had a preliminary investigation of @-mention used in the pull-requests hosted in Ruby on Rails [8]. Extending this prior work, we also 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 to the processing of pull-requests in GitHub. But in GitHub, general issues and pull-requests are both managed by the issue tracking system, we believe it would be interesting to expand these studies by considering the @-mention used in total issues. Different from our previous work, we put forward a comprehensive analysis of the @-mention in both general issues and pull-requests in this study, also we explore the @-network in this investigation involving its network structure, its evolution as well as its potential to mine the relationships and characteristics of developers in the software development.

3 Background & experimental setup

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

3.1 Issues

Reporting issues (bugs, new features or requirements) may be the most common one to contribute among the different ways (e.g., testing, coding, etc.) [23]. These issues are managed by the issue tracking system, which aim at facilitating the management of issues, by providing a feature-rich interface. Generally, issue consists of 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 in issue tracking, such as filing issue tickets, labeling them, setting the milestone and submitting the comments. In our study, we divide the issues into two kinds, general-issues and pull-requests. As the examples shown in Figure 1, issue #16831 is a general-issue and issue #18936 is a pull-request. There are some differences between the interfaces of general-issues and pull-requests 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]. It is also maintained by the issue tracking system in GitHub. For each opened pull-request,

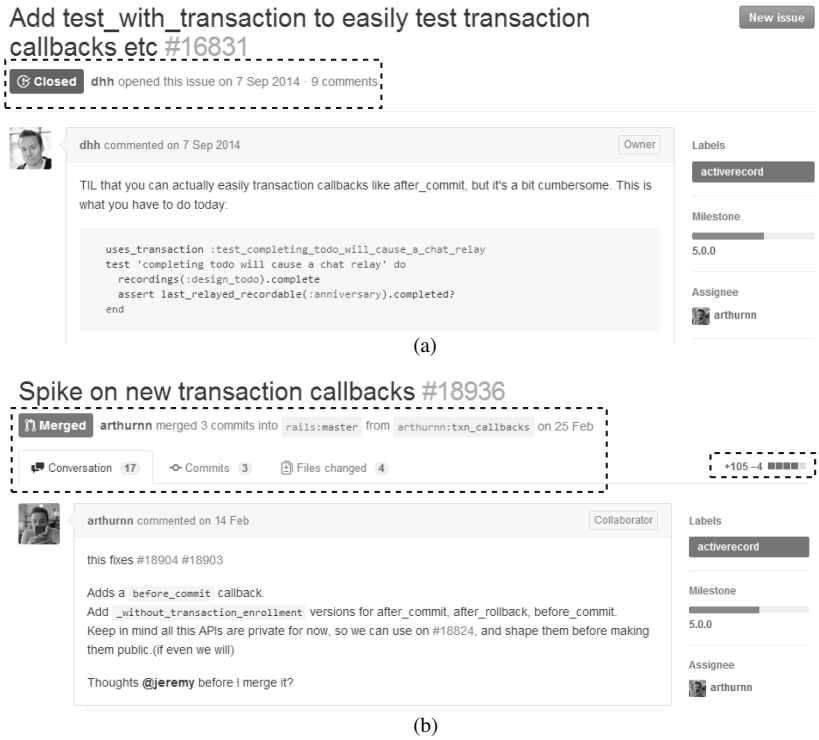


Figure 1 Two examples of issues in GitHub. (a) General-issue; (b) pull-request.

an issue is opened automatically. So every pull-request is an issue, but not every issue is a pull-request. We can consider the pull-requests as special issues in addition to the general-issues. Different from the general-issues, pull-requests have other two types of comments beside general comments (basic type of comments on general-issues or pull-requests): (1) pull-request review comments (comments on the portion of diff patch in pull-requests) and (2) commit comments (comments on the commits of pull-requests).

3.2 @-mention

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

3.3 Research questions

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

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

For answering this question, we choose two famous and large projects hosted on GitHub for our investigation. We analyze the usage of @-mention in the issues of the two projects, including the usage

5) http://en.wikipedia.org/wiki/@.

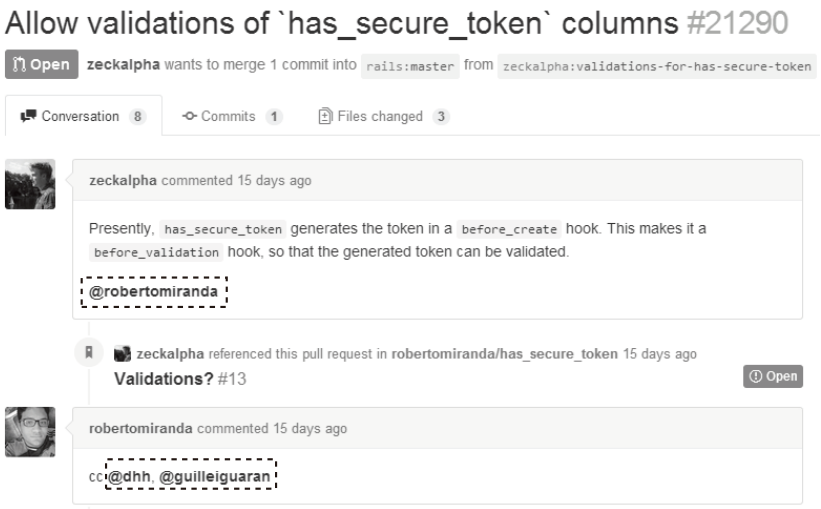


Figure 2 An example of @-mention used in GitHub.

Table 1 The 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

frequency, usage location, usage scenarios as well as the usage participants etc.

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

In answer to this question, we mainly focus on comparing differences between issues with @-mention and without @-mention on the basis of the following characteristics of issues: the number of comments, the number of participants, the description complexity and the time to solve etc. Then we use the 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?

To answer this question, we analyze the basic network structural properties of the @-network and generate many snapshots to capture the evolution of @-network: update frequency and network structure change. Then we run the PageRank algorithm in the @-network to identify the influential developers.

3.4 Research data set

We perform our empirical study on two famous projects Ruby on Rails[6] (Rails) and AngularJS[7], which are both 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. Rails is also one of the great open source projects that GitHub is using to power its infrastructure. AngularJS is a toolset for building the framework most suited to web application development. Table 1 shows the basic information of Rails and AngularJS up to August 2015.

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 collect 21273 issues from Rails and 12647 issues from AngularJS. The specific information of our research database is shown in Table 2. Where "#" means the number of.

In order to analyze the @-mention, we need to extract the information of @-mention (e.g., the @-mentioned developer, the @-mention location etc.) from the textual data of issues (description body and

6) https://github.com/rails/rails.
7) https://github.com/angular/angular.js.
8) http://developer.github.com/.

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

comments). As shown in Algorithm 1, the extraction work can be divided into 4 steps: (1) For each issue in project, 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 its pull-request review comments and commit comments and add them into the issueText, otherwise we directly go to (3). (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, i.e., the username in the string "@username", otherwise we scan the next issue. (4) We query the Project Users table to check whether the "@" is a valid @-mention action. Because some text in back of "@" are not real username, such as "Hongli Lai ⟨hongli@phusion.nl⟩", which is an email address. If it is a valid @-mention action, then we insert the valid @-mention information into our MySQL database for the manipulation in the subsequent phases, otherwise we scan the next issue.

Algorithm 1 Extracting @-mention method

```
1: Input: Issues //general-issues or pull-requests
2: @-mention database ← ∅
3: foreach issue I_a in Issues do:
4:     descriptionText ← extractDescription(I_a)
5:     generalCommentsText ← extractGeneralComments(I_a)
6:     issueText ← descriptionText ∩ generalCommentsText
7:     if I_a ∈ pull-requests:
8:         pullRequestReviewCommentsText ← extractPullRequestReviewComments(I_a)
9:         commitCommentsText ← extractCommitComments(I_a)
10:        issueText ← issueText ∩ pullRequestReviewCommentsText ∩ commitCommentsText
11:    if @ ∈ issueText:
12:        atDeveloperData ← extractAtDeveloperData(issueText)
13:        foreach developer D_a in atDeveloperData do:
14:            if D_a ∈ Project Users:
15:                @-mention database ← {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 that were provided in Subsection 3.3.

4.1 RQ1: @-mention usage

First, we try to investigate the usage of @-mention. In our study, we separately analyze the usage frequency, usage location, usage scenarios as well as usage participants in Rails and AngularJS.

4.1.1 *Usage frequency*

After parsing the issueText by the Algorithm 1, we extract 41395 @-mention actions in Rails project and 15668 @-mention actions in AngularJS project. We find that 11124 issues (52% of 21273 issues) contain

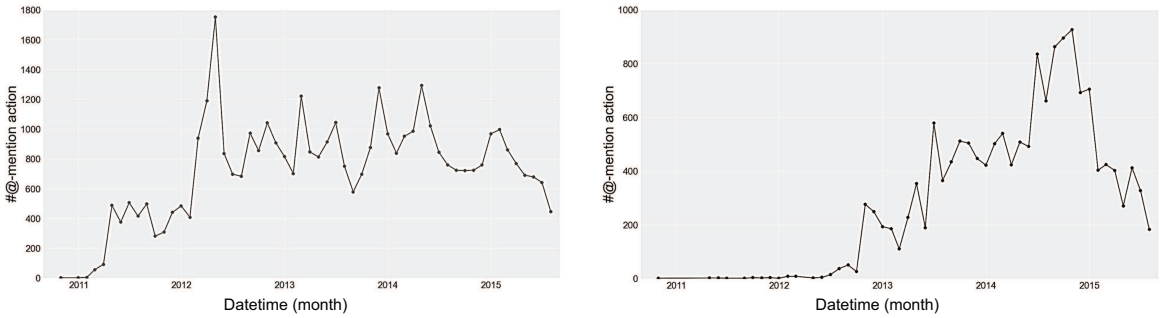


Figure 3 The number of @-mention action occurred in every month in Rails and AngularJS. (a) Rails; (b) AngularJS.

Table 3 The number of @-mention 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

@-mention in Rails and 5332 issues (42% of 12647 issues) contain @-mention in AngularJS. Figure 3 shows the number of @-mention action occurred in every month in Rails and AngularJS. On average, in each month, there are 726 @-mention actions occurred in Rails and 307 @-mention actions occurred in AngularJS. This result reveals that @-mention is used quite a lot in the large and famous projects, which provides enough data set for our investigation.

Actually, each issue may contain many @-mention actions. So we study how many @-mention on average are used in each issue that has @-mention action. Table 3 shows the main statistical results. The first row (#@-mention) shows the number of @-mention action used in each issue. The second and third rows show the specific number of issues (ratio) in Rails and AngularJS. We can find that the vast majority of issues (Rails: nearly 65%, AngularJS: nearly 75%) only have 1 to 3 @-mention actions. This result reveals that the frequency of each issue using @-mention is low.

4.1.2 *Usage location*

As mentioned in Subsections 3.1 and 3.2, @-mention is usually used in the issue's description body or general comments. When in pull-requests, @-mention is also used in pull-request review comments or commit comments. So we study the specific location of @-mention used in issues. Figure 4 shows that, in Rails, 94% @-mention are used in the comments (88% in general comments) and only 6% @-mention are used in the description body. In AngularJS, 98% @-mention are used in the comments (95% in general comments) and only 2% @-mention are used in the description body. This result indicates that most @-mention are used in issue's comments (especially in the general comments) rather than issue's description body, i.e., @-mention is more likely to be used in the conversation of developers.

4.1.3 *Usage scenarios*

In our study, we divide the scenarios of @-mention used in issues into two kinds, "@ submitter" and "@ reviewer". "@ submitter" means the reviewer participated in the issue's discussion @ the issue's submitter and "@ reviewer" means the issue's submitter @ the reviewer or the reviewers @ each other. We find that 24% @-mention are used for "@ submitter" while 76% @-mention are used for "@ reviewer" in Rails. And in AngularJS, 31% @-mention are used for "@ submitter" while 69% @-mention are used for "@ reviewer". This result indicates that most @-mention are used by the issue's submitters to @ other developers for review or reviewers @ each other for discussion.

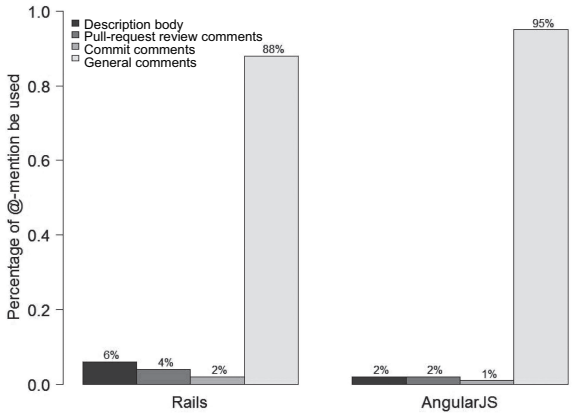


Figure 4 The distribution of specific location of @-mention used in issues.

4.1.4 *Usage participants*

We find that among the total 10094 developers in Rails, 10057 of them are not internal project members but also contribute their codes and suggestions to the only 37 internal project members. In AngularJS, 9699 external developers contribute to the only 35 internal project members. The internal project members can access to the repository of project and they manage all the issues from internal or external developers.

Furthermore, we investigate the participants of @-mention action. We divide the participants of @-mention used in issues into two kinds, "@-mentioning developers" and "@-mentioned developers". "@-mentioning developers" means the developers who are @-mentioning other developers and "@-mentioned developers" means the developers who are @-mentioned by others. In our study, we find that in Rails, 43% "@-mentioning developers" and 54% "@-mentioned" developers are internal project members. Each internal project member on average can @ other developers 481 times and be @-mentioned 606 times, while each external developer on average only @ other developers 3 times and be @-mentioned 2 times. In AngularJS, 55% "@-mentioning developers" and 51% "@-mentioned developers" are internal project members. So each internal project member on average can @ other developers 246 times and be @-mentioned 228 times, while each external developer on average only @ other developers 1 time and be @-mentioned 1 time. This result proves that @-mention is generally used in the group of internal project members.

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

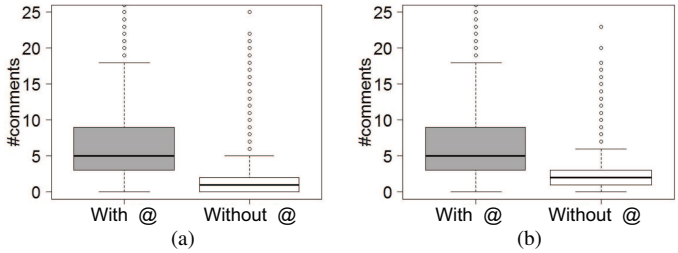


Figure 5 The number of comments in issues with @-mention and without @-mention[9]. (a) Rails; (b) AngularJS.

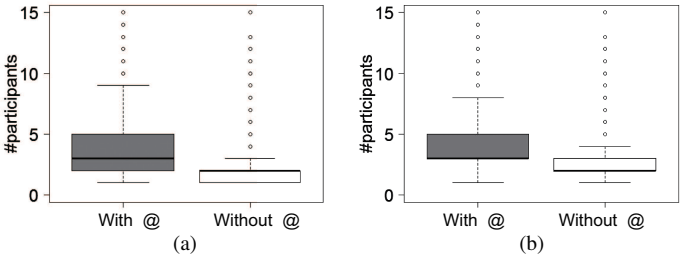


Figure 6 The number of participants in issues with @-mention and without @-mention. (a) Rails; (b) AngularJS.

4.2 RQ2: @-mention influence

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

4.2.1 *The number of comments*

First, we study the distribution of the number of comments in issues with @-mention 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, while the value is raised 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, while the value is raised to 7.7 (median: 5.0) for issues with @-mention.

Using the statistical tests, we verify that the differences between issues with @-mention and without @-mention are statistically significant (Rails: W=94755000, p<2.2E−16, z=62.8 [very significant], δ=0.68 [large]; AngularJS: W=32326000, p<2.2E−16, z=36.0 [very significant], δ=0.66 [large]). This result indicates that issues with @-mention are likely to have more comments than issues without @-mention, i.e., @-mention promotes the discussion of developers in issues.

4.2.2 *The number of participants*

Then, we study the distribution of the number of participants (submitter and reviewers) in issues with @-mention 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, while the value is raised 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, while the value is raised to 4.2 (median: 3.0) for issues with @-mention.

9) In the boxplot, there are 5 main horizontal lines. The top line indicates the max value. The second line indicates the upper quartile. The third line indicates the median value. The fourth line indicates the low quartile. 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).

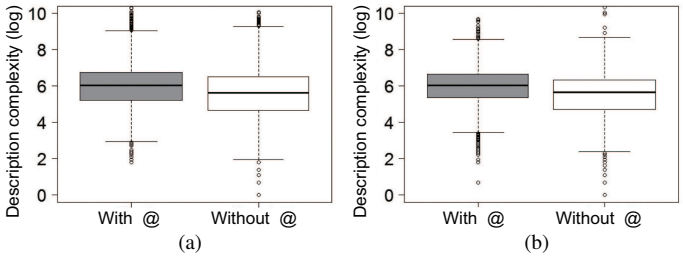


Figure 7 The description complexity in issues with @-mention and without @-mention. (a) Rails; (b) AngularJS.

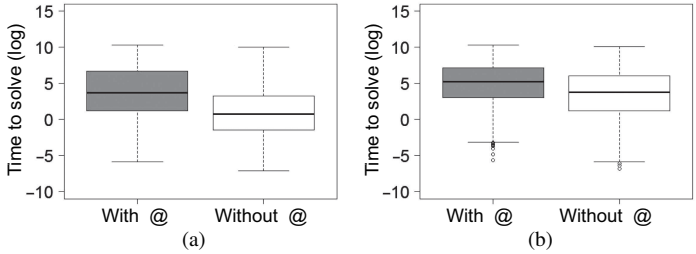


Figure 8 The time-to-solve in issues with @-mention and without @-mention. (a) Rails; (b) AngularJS.

We test and confirm that the distributions between issues with @-mention and without @-mention are significantly different using the statistical tests (Rails: W=93673000, p<2.2E−16, z=71.5 [very significant], δ=0.66 [large]; AngularJS: W=31022000, p<2.2E−16, z=41.7 [very significant], δ=0.59 [large]). This result indicates that issues with @-mention are likely to have more participants than issues without @-mention, i.e., @-mention facilitates developers to participate in the solving process of issues.

4.2.3 *The description complexity*

Then we study the description complexity (the textual length of title and issue's description body) of issues with @-mention and without @-mention. Figure 7(a) shows that in Rails, the average description complexity is 618.3 (median: 276.0) for issues without @-mention, while the value is raised 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, while the value is raised to 639.0 (median: 424.0) for issues with @-mention.

Taking the statistical tests, we find that the differences between issues with @-mention and without @-mention are basically statistically significant except the value of cliff's δ (Rails: W=66571000, p<2.2E−16, z=7.2 [very significant], δ=0.18 [small]; AngularJS: W=23824000, p<2.2E−16, z=10.0 [very significant], δ=0.22 [small]). This result indicates that issues with @-mention may have more description text than issues without @-mention, i.e., issues with much description are more likely to use @-mention than issues with less description.

4.2.4 *The time to solve*

Then we study the distribution of the time-to-solve (time interval between a issue is opened and closed) in issues with @-mention and without @-mention. In our study, we only focus on the closed issues. Figure 8(a) shows that in Rails, the average time to solve the issues with @-mention is 1330.7 hours (median: 40.6 hours), while the time is dropped to 312.0 hours (median: 2.2 hours) for issues without @-mention. Figure 8(b) shows that in AngularJS, the average time to solve the issues with @-mention is 1463.5 hours (median: 186.8 hours), while the time is dropped to 1143.1 hours (median: 42.0 hours) for issues without @-mention. In our statistical tests, the results prove that these differences are basically statistically significant (Rails: W=73391000, p<2.2E−16, z=31.2 [very significant], δ=0.42 [medium]; AngularJS: W=19115000, p<2.2E−16, z=5.5 [very significant], δ=0.23 [small]). This result indicates that issues with @-mention are likely to need more time to deal with than issues without @-mention.

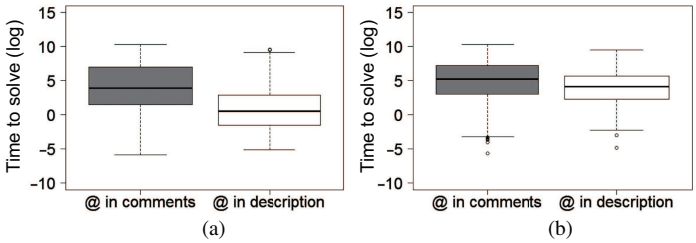


Figure 9 The time-to-solve in issues with @-mention only in description body and only in comments. (a) Rails; (b) AngularJS.

Table 4 Other characteristics of issues with @-mention 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 ***

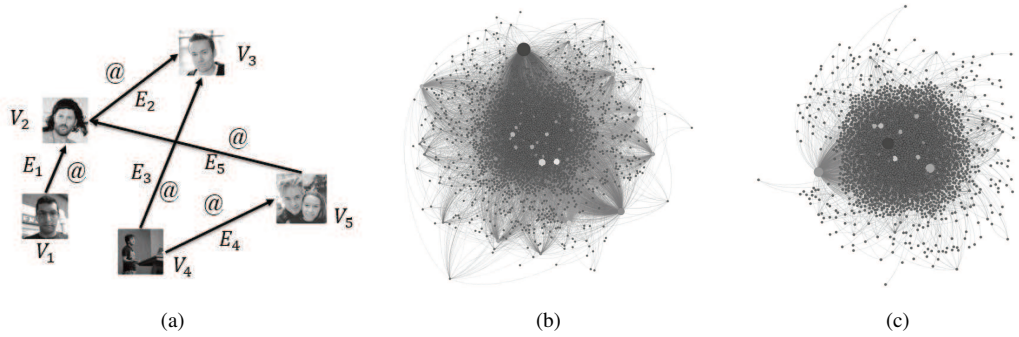
'***' p<0.001, '**' p<0.01, '*' p<0.05, '.' p<0.1

As mentioned in before, @-mention can be used in the issue's description body or comments. So we want to analyze the impact of specific location of @-mention to the time-to-solve. We filter out two groups from the closed issues: issues with @-mention only exist in they description body and issues with @-mention only exist in they comments. Figure 9(a) shows that in Rails, the average time to solve the issues with @-mention only in description body is 198.7 hours (median: 1.7 hours), while the time is raised to 1462.6 hours (median: 50.2 hours) 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 hours (median: 62.3 hours), while the time is raised to 1498.2 hours (median: 190.8 hours) for issues with @-mention only in comments. In our statistical tests, the results prove that these differences are basically statistically significant (Rails: W=4527300, p<2.2E−16, z=24.0 [very significant], δ=0.50 [large]; AngularJS: W=252510, p<2.2E−16, z=3.7 [very significant], δ=0.23 [small]). This result indicates that @-mention in description body has more impact on the solving process of issues than @-mention in comments.

4.2.5 *Other characteristics of issues*

We also analyze other characteristics (label, milestone, assignee and state) of issues with @-mention and issues without @-mention. As shown in Table 4, we find that in Rails, 43.9% issues with @-mention have label while the ratio for issues without @-mention is only 14.2%. In AngularJS, 71.5% issues with @-mention have label while the ratio for issues without @-mention is only 53.0%. Similarly, issues with @-mention are more likely to have 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 @-mention and without @-mention are significant. These results reveal that issues with @-mention are more likely to have label, milestone as well as assignee than issues without @-mention. As mentioned before, issues with @-mention need more time to solve than issues without @-mention. So both in Rails and AngularJS, the ratio of closed issues in issues with @-mention is little less than issues without @-mention.

Discussion. By using statistical tests, we find that issues with @-mention may have more comments, more participants, more description text as well as longer time-to-solve than issues without @-mention, which indicates that @-mention is very likely to be used in the difficult issues. We think that difficult issues need more time and resources to deal with but the mechanism of @-mention can promote the discussion of developers in issues and facilitate developers to participate in the solving process of issues.



(a) (b) (c)

Figure 10 Building @-network in Rails and AngularJS. (a) An example of @-network; (b) @-network in Rails; (c) @-network in AngularJS.

We further find that @-mention in issue's description body have more effect on the time-to-solve of issues than @-mention in comments. We explain that @-mention in description body can enlarge the visibility of issues which indicates that this issue is very purpose. We also find that issues with @-mention are more likely to have label, milestone and assignee than issues without @-mention, which reveals that issues with @-mention get more attention and well management than issues without @-mention. These results prove that @-mention help involve more participants and has a positive impact on the solving process of issues.

4.3 RQ3: @-network

During our investigation, we believe that @-mention has many possible research directions. In particular, @-mention builds a social network among the developers in the solving process of issues, which contains rich information for mining and research.

4.3.1 *Building @-network*

We firstly construct a social network based on our @-mention database, which we called @-network. The @-network can be defined as a directed graph $G_{fn} = (V, E)$. V represents the set of vertices which are all developers participate in the project. E presents a set of node pairs $E(V) = \{(u, v)|u, v \in V\}$. If the node v_j is @-mentioned by v_i, then there is an edge from v_i to v_j. For example in the Figure 10(a), V_2 is @-mentioned by V_1, so there is a directed edge E_1 from V_1 to V_2. Figure 10(b) and (c) shows the total @-network redrawn by Force Atlas[10] algorithm 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 compute 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), revealing that each developer may @ 1.8 other developers and be @-mentioned 1.7 times. In the @-network of AngularJS, the average degree is 4.2 (indegree: 2.1, outdegree: 2.1). By following Surian et al. [26] and Leskovec et al. [27], the shortest path between two nodes is computed by ignoring the weights of the edges in the graph. The length of a path between two nodes is simply the length of the series of nodes between the two nodes. In the @-network of Rails, the average shortest path is 3.19 and the value is 3.15 in the @-network of AngularJS. Surian et al. [26] studied the Sourceforge[11] project hosting platform and found that the average shortest path among project developers is 6.55, following the popular assumption of "six-degree-separation" [28]. Other study of the Facebook social graph has concluded that individuals on Facebook have potentially tremendous reach with an average shortest path of 4.7 [29]. The average shortest path in @-network is significantly lower, which suggests that the social media tool @-mention actually enables more collaboration among developers. The @-network allows for even better reach as developer's relationships are tighter than human's relationships in daily life social networks.

10) https://github.com/gephi/gephi/wiki/Force-Atlas-2.
11) http://sourceforge.net/.

Table 5 The basic network 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

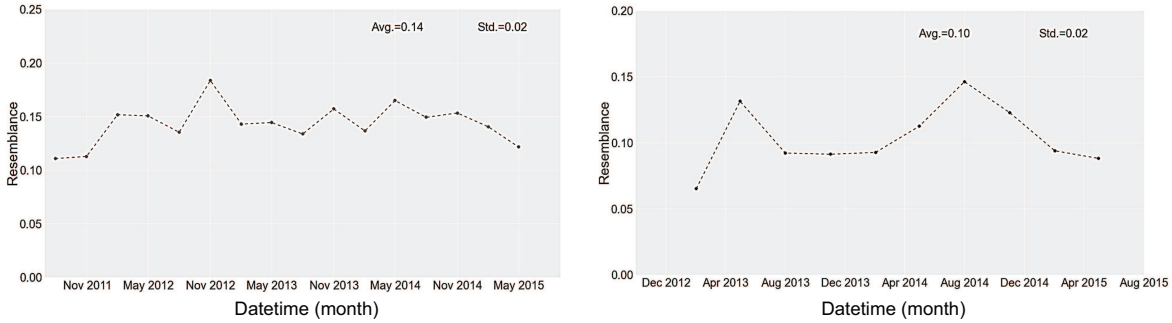


Figure 11 Resemblance of the @-network. (a) Rails; (b) AngularJS.

4.3.2 *@-network evolution over time*

We then study 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 study how the @-network evolves over time by deriving multiple snapshots of the network. We generate snapshots of the @-network based on the @-mention action at 90 day intervals. In Subsection 4.1.1, we can find that only a small amount of @-mention actions occurred before May 2011 in Rails and November 2012 in AngularJS. So in our study, we generate 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 @-mentioning node pairs (v_i, v_j) during the 90 days period before this snapshot. Below, we examine the evolving @-network based on these snapshots.

(a) Update frequency of the @-network. First, we investigate the update frequency of the @-network from one snapshot to the next. Similar to the approach proposed by Viswanath et al. [30], we use 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. Where P_t represents the set of @-mentioning node pairs that are active at time t. The value of R_t varies from 0 to 1. $R_t = 1$ means the entire set of active @-mentioning node pairs continued to interact at the next time period. While $R_t = 0$ means that none of the @-mentioning node pairs who interacted in time t interacted in time $t + 1$.

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

Figure 11(a) shows that in Rails, the average resemblance across all snapshots is 0.14 (Std. = 0.02), which indicates that on 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), which means that only 10% of the @-mentioning node pairs remain active over time and 90% no longer exist in the consecutive snapshot. The result reveals that the @-mentioning node pairs in the @-network change dynamically over time.

Furthermore, we mine the companion existed in the @-network, which companion means the set of @-mentioning node pairs that exist in a long time period (two or three years). Table 6 shows that in Rails, only 11 (0.06%) companions exist at least 12 snapshots (more than three years). In AngularJS, only 4 (0.05%) companions exist at least 8 snapshots (more than two years). This result indicates that only a small fraction of the @-network is stable in a long time period.

Table 6 The 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)

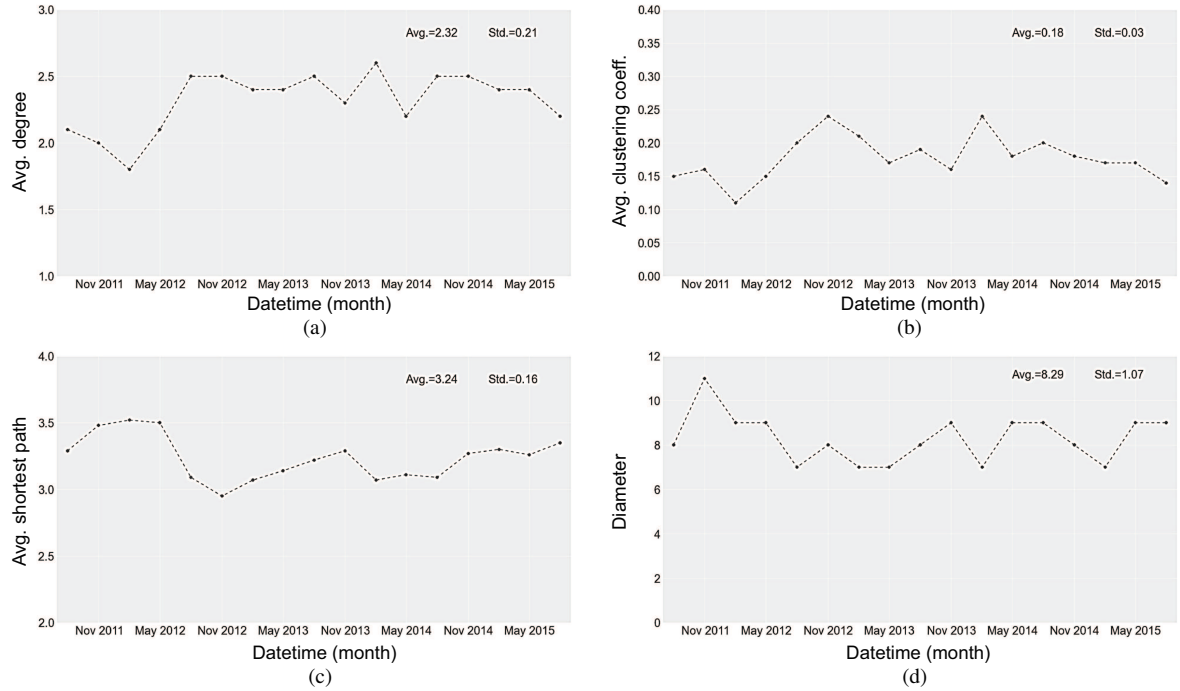


Figure 12 Structural properties of the evolving @-network in Rails. (a) Average degree; (b) average clustering coefficient; (c) average shortest path; (d) diameter.

(b) Network structure change of the @-network. We then investigate to what extent network structural properties of the @-network change over time. To give a description in detail, we calculate 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), the average clustering coefficient is 0.18 (Std.=0.03), the average shortest path is 3.24 (Std.=0.16) and the average diameter is 8.29 (Std.=1.07). And in AngularJS, Figure 13 shows that the average degree is 1.68 (Std.=0.14), the average clustering coefficient is 0.14 (Std.=0.03), the average shortest path is 3.04 (Std.=0.16) and the average diameter is 7.27 (Std.=0.96). By analyzing the four network metrics vary over the snapshots in Rails and AngularJS, we can find that the different network measures are all basically stable over time with some small fluctuations. Based on our finding about the fast 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.

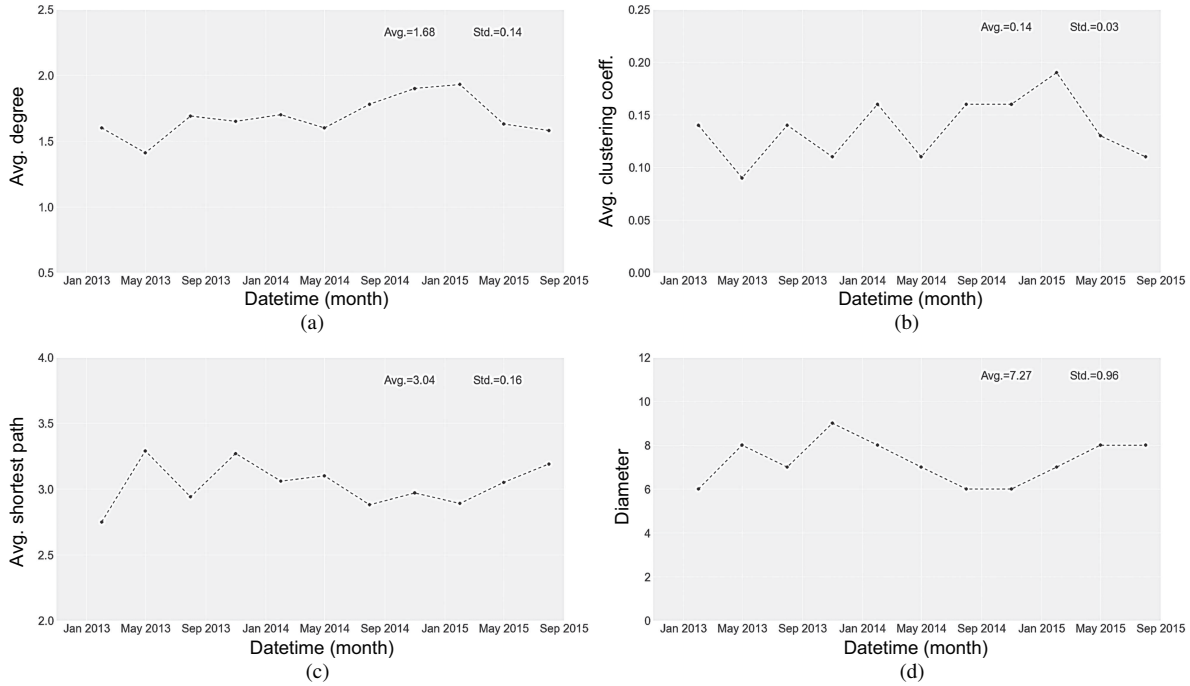


Figure 13 Structural properties of the evolving @-network in AngularJS. (a) Average degree; (b) average clustering coefficient; (c) average shortest path; (d) diameter.

4.3.3 *Identifying the influential developers*

Furthermore, we want to identify the influential developers from the @-network. We run the PageRank algorithm in the @-network. PageRank algorithm, which is for weighting web pages importance based on their links, has gained popularity driven by its use in the Google search engine [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. Then subsequent interactions 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

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

The PageRank algorithm returns a PageRank score for every developer. We can separately get the top-10 influential developers in Rails and AngularJS in terms of their PageRank scores. The results are shown in Tables 7 and 8. 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, which indicates that he is an active developer and has a lot of contribution to the Rails. The top-1 developer in AngularJS is "caitp", who has submitted 263 issues and 6594 comments, which reveals that he is an active and influential developer in AngularJS.

This result reveals that developers who are active or contribute more to the project, get higher PageRank scores than developers who are negative and have less contribution. By analyzing the correlation between the PageRank scores and the characteristics of developers, we find that the characteristics of developers are basically consistent with the PageRank scores, i.e., we can evaluate the contribution, activeness and influence of developers by mining the @-network.

Discussion. @-mention builds a social network among the developers in the solving process of issues,

Table 7 The 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 The 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

which contains rich information for mining. By analyzing this @-network, we find that @-network allows for even better reach as developers' relationships are tighter than the human's relationships in daily life social networks. We believe that all the developers participate in the solving process of issues are very purposeful to solve the issues. Strong purpose enables developers to interact more closely. Through the evolution of @-network, we analyze the refresh rate of @-mentioning node pairs and resulting global network structural properties. The result reveals that while the individual @-mentioning node pairs that constitute the @-network change over time, the average network structural properties remained relatively stable. Furthermore, we find that in the @-network, the PageRank score of developers are basically consistent with their characteristics of software development. This phenomenon proves that we can identify the influential developers in the software development. We certainly have the potential to assist the software development by exploiting the knowledge from the @-network. Nevertheless, more research needs to be done 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 use the number of comments, the number of participants etc. as measurements to verify the impact of @-mention on the issues. Future work is needed on analyzing some other characteristics of issues, e.g., the number of files changed and the code churn in pull-requests. Since @-mention can be used in so many ways, e.g., expression of disagreement or notification, we cannot be sure, without further analysis, whether an @-mention expresses trust, distrust, or none of them. We need take more research to in-depth analyze the specific expression of @-mention in GitHub's issues.

5.2 External validity

We find that there exists some abnormal issues that about simple problem but have a long handling time. Also some difficult issues are solved quickly because they are related to the coming release milestone of the project. Some issues are new feature requirements and some are real bugs. The different types of issues may lead to bias in our study.

6 Conclusion

In this paper, we obtain a deep understanding of the role of @-mention in assisting software development in the issues of GitHub. We examine data from two large and famous projects Rails and AngularJS, including @-mention usage, @-mention influence and the characteristics of @-network. By statistical analysis, we find that @-mention, as a famous social media widely used in online social platform, is used in the software development in GitHub quite a lot too. Most @-mention are used in the issues' comments for "@ reviewer" and they are basically used in the group of internal project members. Furthermore, we find that @-mention is beneficial for involving more collaboration and tends to be used in the difficult issues. We prove that @-mention enlarges the visibility of issues and facilitates the developers' discussion. Also issues with @-mention are more likely to have label, milestone as well as assignee than issues without @-mention. Our investigation shows that @-mention has a positive impact on the solving process of issues. Based on the @-mention database, we also build an @-network for mining the relationships and characteristics of developers. Our analysis indicates that even though there is high churn in the @-mentioning node pairs that interact over time, many of the global structural properties of @-network remain relatively constant over time. Our study proves that we certainly have the potential to assist the software development by exploiting the knowledge from the @-network. Next steps will focus on expanding our investigation around the study of @-network.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant Nos. 61432020, 61472430, 61502512) and 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. Washington: IEEE Computer Society, 2007. 6
2 Bogdan V. Human aspects, gamificaiton, and social media in collaborative software engineering. In: Proceedings of the 36th International Conference on Software Engineering. New York: ACM, 2014. 646–649
3 Ahmadi N, Jazayeri M, Lelli F, et al. A survey of social software engineering. In: Proceedings of the 23rd IEEE/ACM International Automated Software Engineering Workshops, L'Aquila, 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. New York: 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. New York: 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. New York: 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, San Francisco, 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. New York: 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, Oakland, 2010. 2: 1–2
10 Louridas P. Using wikis in software development. IEEE Trans Softw, 2006, 23: 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. Washington: IEEE Computer Society, 2009.

74–77
12 Riemer K, Richter A. Tweet inside: microblogging in a corporate context. In: Proceedings of the 23rd Bled eConference eTrust: Implications for the Individual, Enterprises and Society, Bled, 2010. 1–17
13 Yang J, Counts S. Predicting the speed, scale, and range of information diffusion in twitter. In: Proceedings of International Conference on Weblogs and Social Media, Washington, 2010. 355–358
14 Vega E, Parthasarathy R, Torres J. Where are my tweeps?: Twitter usage at conferences. Paper, Personal Information Management Class, Virginia Polytechnic Institute and State University, 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 Softw Eng, 2009, 35: 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. Washington: IEEE Computer Society, 2004. 387–396
17 Omoronyia I, Ferguson J, Roper M, et al. Using developer activity data to enhance awareness during collaborative software development. Comput Supp Coop Work, 2009, 18: 509–558
18 Kotlarsky J, Oshri I. Social ties, knowledge sharing and successful collaboration in globally distributed system development projects. Euro J Inf Syst, 2005, 14: 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. New York: ACM, 2010. 1–5
20 O'reilly T. What is Web 2.0: design patterns and business models for the next generation of software. Commun Strat, 2007, 65: 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, Istanbul, 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, Jeju, 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, Montreal, 2015. 550–554
24 Thung F, Bissyandé 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. Washington: IEEE Computer Society, 2013. 323–326
25 Gousios G, Pinzger M, van Deursen A. An exploration of the pull-based software development model. In: Proceedings of the 36th International Conference on Software Engineering. New York: ACM, 2014. 345–355
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, Beverly, 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. New York: ACM, 2008. 915–924
28 Travers J, Milgram S. An experimental study of the small world problem. Sociometry, 1969, 32: 425–443
29 Ugander J, Karrer B, Backstrom L, et al. The anatomy of the facebook social graph. arXiv: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. New York: ACM, 2009. 37–42
31 Brin S, Page L. Reprint of: the anatomy of a large-scale hypertextual web search engine. Comput Netw, 2012, 56: 3825–3833