International Journal of Database Theory and Application
Vol. 10, No.12 (2017), pp. 13-22
hhtp://dx.doi.org/10.14257/ijdta.2017.10.12.02

ScaffdCF: A Prototype Interface for Managing Conflicts in Peer Review Process of Open Collaboration Projects

Wenjian Huang, Tun Lu and Ning Gu

School of Computer Science, Fudan University, Shanghai, China
Shanghai Key Laboratory of Data Science, Fudan University, Shanghai, China
{wenjianhuang11, lutun, ninggu}@fudan.edu.cn

Abstract

Open collaboration projects (e.g., Wikipedia, open-source software projects) usually employ peer review mechanism to ensure the quality of productions. Due to the large variance of contributors and reviewers' backgrounds, expertise, and interests, conflicts are often unavoidable. Existing peer review systems do not support effective conflict management. This paper presents a prototype interface, named "ScaffdCF", for project reviewers and administrators to manage conflict more effectively. We identify three problems of existing peer review systems: 1) argument points are overwhelmed; 2) conflict management guidelines are lacking; and 3) contexts about members in conflict are lacking. To address these problems, ScaffdCF integrates new features to scaffold conflict management process. A survey-based user study shows that some features (e.g., highlighting arguments, explicitly expressing agreement or not) can help to manage conflict more effectively without significantly increasing administrators' cognitive overload. Meanwhile, some features (e.g., decomposition into sub-issues) fail. We discussed the limitations and future improvements.

Keywords: *conflict management; peer review; open collaboration; prototype interface*

1. Introduction

In open collaboration projects (*e.g.*, Wikipedia, open-source software development projects), people with varied backgrounds and expertise collaborate to create knowledge or artifacts. To ensure the quality of productions, peer review is commonly adopted. For instance, in GitHub – the largest open-source software development platform, a member's code changes should be submitted as a "pull-request" and fully reviewed by other members of the project [12]. Only pull-requests without any problem can be accepted and merged into the main code repository.

Due to the large variance of members' backgrounds, expertise, and interests, conflicts between contributors and reviewers are often unavoidable, especially when reviewers rejected one's contribution. Prior studies in the context of traditional organizations suggested that conflicts generally had negative effects on group loyalty, work productivity, and job satisfaction [2, 6, 8]. A recent research found that the argument between contributors and reviewers in GitHub would make the likelihood of a contributor leaving the project increase by 16.8% [19].

This paper aims to design a prototype interface, named "ScaffdCF", which provides various scaffolding features for project reviewers or administrators to manage conflicts more effectively. We identified several problems of existing peer review process and design new features accordingly to overcome these problems.

Received (October 9, 2017), Review Result (November 24, 2017), Accepted (December 5, 2017)

ISSN: 2005-4270 IJDTA

First, since the discussions between contributors and reviewers are heterogeneous and multifaceted, the focal argument points might be overwhelmed in the tedious post threads. To overcome this problem, a feature is designed to highlight arguing comments. It is also supported for users to establish sub-issues for a contribution and re-organize these sub-issues by tabs. Besides, project members can explicitly express their agreements or disagreements on a sub-issue.

Secondly, when conflict occurs, few explicit guidelines or tips suggest which way to communicate with members in conflict is more effective. In our prototype, when a contributor's argument is detected, a tip encouraging reviewers to give concrete advices to solve the problem is displayed and reviewers should also check whether their comments indeed have any advices.

Thirdly, some contexts about members in conflict (*e.g.*, how many times a contributor argued before, how often a reviewer gave criticisms before) are critical for the decision making about how to deal with the conflict. So, in our prototype, some basic and historical information (i.e., background, prior experience, and prior arguments) about contributors and reviewers is given on the page.

A survey-based user study was conducted to examine the effectiveness of ScaffdCF. The result shows that some features (*e.g.*, highlighting arguments, explicitly expressing agreement or not) are helpful for effective conflict management without significantly increasing administrators' cognitive overload. Meanwhile, some features (*e.g.*, decomposition into sub-issues) fail. We discuss the limitations of ScaffdCF and possible improvements on it.

The rest of this paper is organized as follows. Section II reports some related work about conflict and its management in open collaboration projects. In section III, we analyze the disadvantages of existing peer review process for conflict management. In section IV, we present the implementation of our prototype interface. Section V presents the user study.

2. Related Work

2.1. Conflict in Open Collaboration Projects

In open collaboration projects where a number of participants have to combine their own agendas and points of view, conflict is likely unavoidable. Rahim described conflict as "an interactive process manifested in incompatibility, disagreement, or dissonance within or between social entities (*i.e.*, individual, group, organization)" [14]. Conflicts generally develop from scenarios such as disagreement between users, procedures and rules for coordination and resolution.

Open collaboration projects are especially prone to conflict due to factors like a lack of shared context, difficulties in sharing information, and reduced familiarity with other members [7]. Kittur *et al.*, described how conflicts arose in Wikipedia when people edit the same parts of articles [9]. In open-source software projects, researchers have found that communication channel issues, information overload, competing technologies and incompatible software versions can often lead to conflicts [5]. Peer review is a common mechanism to ensure the quality of online productions. Conflict might arise in the peer review process when one's contribution was unjustly rejected [18].

In offline settings, it has been proven that conflict generally have negative effects on group loyalty, workgroup productivity, and job satisfaction [2, 6, 8]. Deutsch found that conflicts decrease goodwill and mutual understanding, which hinders the completion of organizational tasks [3]. Haq also suggested that conflict may increase employees' stress in workplace and lead to deviant behavior [6].

2.2. Conflict Management Strategies

Conflict management has been studied from a variety of different perspectives, mostly in the context of traditional organizations. For example, Blake and Mouton proposed five methods to handle conflicts: forcing, withdrawing, smoothing, compromising, and problem solving [1]. Rahim and Bonoma also summarized five styles of handling conflict (*i.e.,* integrating, obliging, dominating, avoiding and compromising) and the situations in which these are appropriate [14]. Thomas described a framework to manage conflict by the degree to which individuals attempt to satisfy their own concerns ("assertiveness") vs. others' concerns ("cooperativeness") [17]. A number of other studies independently attempt to build typologies of conflict management strategies and identify aspects of conflict management that are important to outcomes. Despite these studies, few studies apply these strategies to the design of peer review processes or systems in open collaboration projects for effective conflict management.

3. Problems of Existing Peer Review Process

In this section, we analyze the problems of existing peer review process. Most open collaboration projects rely on a number of volunteers' contribution. The goal of peer review is to decide whether a contribution should be accepted.

A typical instance is open-source software projects' code review mechanism. In GitHub, project members can "fork" the project's code repository, and commit code changes to the forked branch repository. If they want to contribute these changes back to the root repository, they can submit a "pull-request" (PR) to ask for merging changes into the root repository. Then, there is a peer review process. Reviewers can be the project's administrators as well as other peer members. The reviewing process is organized similar to a thread of an online forum. The PR-contributor discusses with reviewers back and forth about the PR's usefulness and any technical issues. Sometimes, reviewers might give advices to further modify the PR. Figure 1 shows the workflow of GitHub's code review process.

Another instance is Wikipedia's open editing model. It can be viewed as an informal peer review mechanism where all contributions are initially accepted and then other editors perform review and reject unwanted contributions [16]. The article's talk page is where editors discuss whether an edit should be reverted.

Existing peer review mechanism is constituted by a simple discussion process back and forth between contributors and reviewers. While this works well for general use, it has some drawbacks when conflict occurs.

Problem 1: The argument points are overwhelmed.

Existing peer review process is organized like a discussion thread in an online forum. As comments grow rapidly, the original arguing comments would be overwhelmed in the tedious discussion thread. This problem is more significant for third-party reviewers, who are not involved in the conflict directly but join the discussion to manage conflict. These third-party reviewers usually didn't join the discussion at first. So, when conflict occurs, they need to quickly learn about the exact argument points. A prior research on open-source software

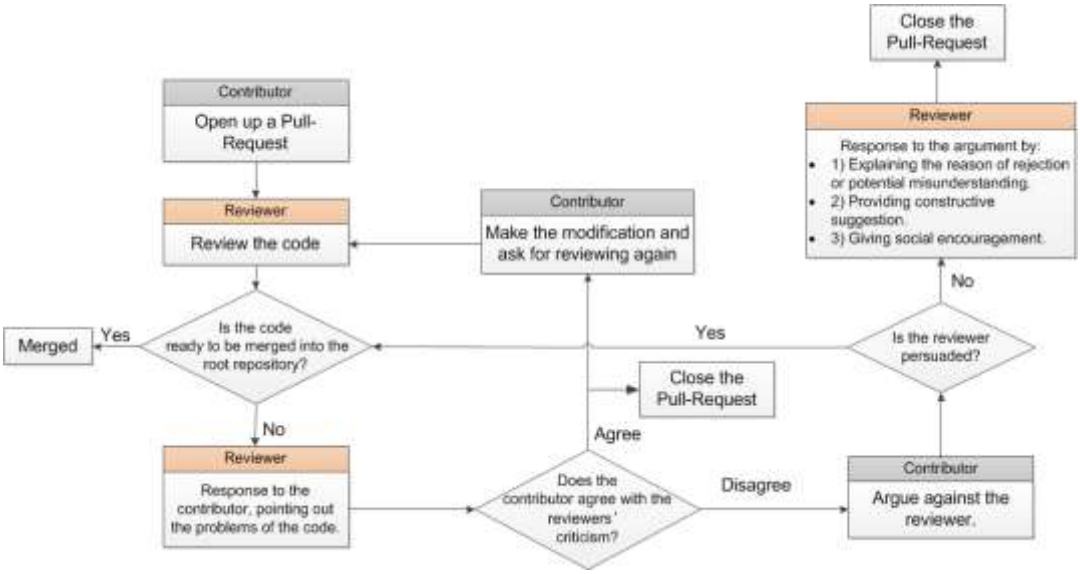

Figure 1. The Workflow of GitHub's Pull-Request Reviewing Process

development projects found that information overload was a heavy burden for developers, because they had to handle a steady flow of communication coming through the mailing lists every day [5]. Similarly, in peer review process, the overwhelming reviews might make reviewers get lost in the noise and fail to find the exact argument points.

Problem 2: Conflict management guidelines are lacking.

Once conflict arose, proper interventions to manage the conflict should be given. However, most existing peer review mechanism does not have explicit guidelines or tips suggesting reviewers which ways to manage conflict are more effective. In traditional organizations, conflict management requires strategic diagnoses and interventions. For instance, Blake and Mouton first presented a conceptual scheme for classifying the styles for managing conflicts into five types: forcing, withdrawing, smoothing, compromising, and problem-solving [1]. Rahim also summarized five specific styles of handling conflict: integrating, obliging, compromising, dominating, and avoiding [14]. In the setting of online open collaboration projects, smoothing and problem-solving are two applicable strategies. A recent study based on GitHub's open-source software projects found that problem-solving (*i.e.*, giving the concrete suggestions to fix the issues) was the only effective strategy to retain arguing contributors to keep participating in projects. Smoothing strategy, that is giving rational explanations or social encouragements, is not effective [19]. Based on these findings, we argue that existing peer review process should be improved by educating reviewers to adopt proper intervention strategies to manage conflict.

Problem 3: Contexts about members in conflict are lacking.

To correctly adjudicate conflict, contexts about members in conflict (*e.g.*, prior experiences in projects) are needed. For instance, if a reviewer in conflict trends to give harsh criticisms on minor issues, which can be learned from his/her prior reviewing experiences, then third-party reviewers should pay more attention to examine whether the conflict is caused by the reviewer's unfair criticism. There have been many theories suggesting that context awareness and social transparency are critical for interactions in distributed collaboration systems. Erickson and Kellogg introduced the concept of social translucence and argued that it is possible to design digital systems that support coherent behavior by making participants and their activities visible to one another [4]. They suggested three characteristics of "socially translucent systems" –visibility, awareness,

and accountability –which enable people to draw upon their social experience and expertise to structure their interactions with one another. Making co-workers more visible and letting them know others' activities in the joint project would encourage participation and promote collaborative work [15]. In the scenario of peer review, it is obvious that being aware of other members' historical activities in the project can help people make proper decisions to deal with the conflict.

4. Implementation of the Prototype Interface

In response to the above problems, we implement a prototype interface for peer-review in open collaboration projects. The prototype is designed based on GitHub's pull-request review model. As we have introduced, in GitHub projects, a developer contribute code changes by submitting a pull-request, and other members review it to decide whether it should be accepted (see Figure 1).

The prototype interface is implemented as a web application, which is built on Node.js framework. We use Angular.js and Bootstrap to build the front-end interface. The back-end data is stored in a MySQL database. Figure 2 shows the main page of the prototype interface, which specifically is the peer-review page about a pull-request. In the following, we present the features designed in response to the above problems of existing peer-review process.

4.1. Features in Response to Problem 1

Problem 1 concerns that the argument points might be overwhelmed in the tedious discussion thread. To overcome this problem, we designed three features.

Highlighting Arguments. In traditional peer review process, few explicit clues infer who is making arguments against someone. Actually, it's feasible to detect contributors' or reviewers' arguing comments, either in a manual manner or through an automatic text mining technique. In our interface, when a user edits his/her comment, he/she can explicitly label the comment as an argument, by choosing an option in the "argument" dropdown button (see Figure 2-E). In such a way, the comment would be labelled with "argument" (see Fig.2-D). However, this manual manner might not be preferred by some users, because people tend to not be too rude to other users. So, a more feasible method to highlighting argument is using automatic text mining techniques. Although the comprehensive text mining techniques to detect arguing comments are not within the scope of this paper, previous studies have proposed various text mining algorithms or models to detect arguments or disputes (*e.g.*, [10, 13]).

Decomposition into sub-issues. For some pull-requests, reviewers might discuss multiple issues of it. If all comments flow in a single thread, it might increase the cognitive overload to find out the comments about a particular issue. So our prototype interface support users to divide the discussion about a pull-request into multiple sub-issues. A user (either the contributor or a reviewer) can create a sub-issue and give the detailed description about this sub-issue with less than 140 words. As Figure 2-B shows, the interface would create an individual tab for this sub-issue. Any reviews about this sub-issue should be carried out in the page of this tab.

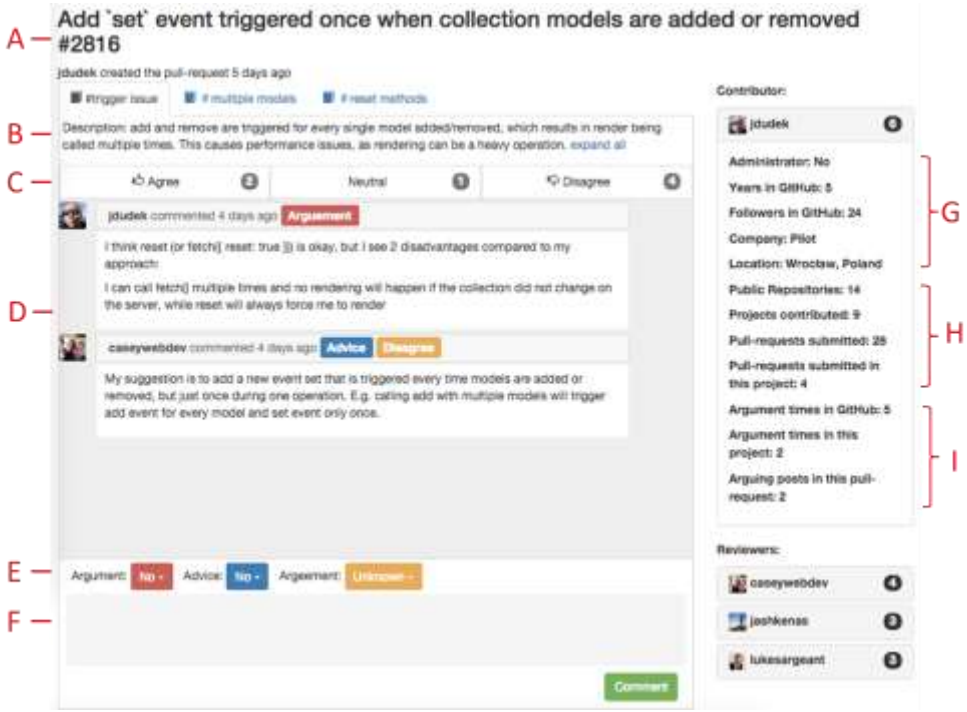

Figure 2. Screenshot of the Main Page of the Prototype Interface. (A:Pull-Request Title; B: Tab and Description for Sub-issues; C: Agreement Buttons; D: Comments; E: Comment Edit Options; F: Comment Edit Areas; G: Background Information; H: Prior Experience; I: Prior Arguments)

Explicitly Expressing agreement or not. For third-party reviewers who come to deal with conflict, they usually need to go through all comments and fully comprehend the exact opinions of each reviewer or contributor. However, in existing peer review process, it is common that a reviewer gives a detailed and tedious comment on a pull-request, which make it difficult for others to comprehend the reviewer's exact opinion towards the pull-request (*i.e.*, does the reviewer agree that the pull-request is problematic or not). So, in our prototype interface, reviewers can explicitly express their attitudes towards a sub-issue by clicking the "agree", "neutral" or "disagree" button (See Figure 2-C). For example, if a reviewer agrees that the sub-issue, as the description states, is true and need to be fixed, then he/she can click the "agree" button. The numbers of reviewers for these three attitudes are also visible. When the mouse cursor hovers over an attitude button, the corresponding reviewer names holding this attitude would be shown. In addition, when writing a comment, the reviewer can also explicitly express his/her attitude towards the sub-issues, by choosing an option of the "agreement" dropdown button. In such a way, third-party can directly comprehend the exact attitudes of reviewers or comments.

4.2. Features in Response to Problem 2

Problem 2 concerns that there is no guidelines for reviewers and project administrators to manage conflict appropriately. To overcome this problem, we designed features to encourage reviewers to give the arguing members with concrete advices to fix the issues.

Promoting Advices. Previous literatures suggest that when dealing with conflicts in peer review process, giving members in conflict with concrete advices to fix the issues is far more effective than other conflict management strategies such as rational explanation or social encouragement [19]. According to this reasoning, we designed features to promote giving advices. When the system detected the occurrence of conflict (*i.e.*, a

contributor argued against a reviewer), a tip saying that "Since an argument arose between the contributor and a reviewer, it is the best choice for you to give advices to help fix their problems" will display below the comment edit box (it is not shown in Figure 2). Besides, if a reviewer did give his/her advices, he/she can label the comment by choosing the "yes" option of the "advice" dropdown button (see Figure 2-E).

4.3. Features in Response to Problem 3

Problem 3 concerns that the contexts about contributors and reviewers are lacking. In our prototype interface, we provide three categories of context information, as shown in Figure 2.

Background information. The first category of context is the contributor or reviewer's background information, such as location, followers count and company (see Figure 2-G). These contexts provide clues inferring the contributor or reviewer's reputation and expertise. For instance, a reviewer who has many followers usually holds a high reputation in the community [11]. A reviewer who is an administrator of the project should has higher expertise on this project.

Prior experience. Another category of context is the reviewer or contributor's prior work experience in the community, such as years joining the community, the number of pull-requests submitted in the whole community and the project, the number of public code repositories (see Figure 2-H). These contexts can help third-party reviewers to look over the prior contributions of members in conflict so as to properly adjudicate conflict.

Prior Arguments. Thirdly, to put the contributor and reviewers' prior arguments into context, we provide three kinds of information: argument times in GitHub, argument times in this project, arguing posts in this pull-request (see Figure 2-I). Prior argument experience can help third-party reviewers to judge whether one's argument or criticism is fair.

5. Survey-Based User Study

To evaluate the effectiveness of ScaffdCF in manage conflicts, we conducted a survey-based user study. In the study, we compared ScaffdCF with GitHub's existing peer review interface. The evaluation is two-fold. One goal is to examine whether ScaffdCF can significantly improve the effectiveness for reviewers to manage conflict. The other goal is to examine whether ScaffdCF significantly increase the cognitive overload of reviewers. We hypothesize that ScaffdCF can significantly improve the effectiveness to manage conflict, but does not significantly increase reviewers' cognitive overload.

The user study is designed based on an online survey. The survey participants are contributors for GitHub's popular open-source software projects. We obtained their email addresses from their profiles on GitHub and sent the survey invitations to them. The online survey was designed via Google Form. In the survey, we presented the interface and explained the features in detail. Then, for each feature, the survey participants were asked whether it can improve the effectiveness for them to manage conflict and whether it would increase their cognitive overload to give reviews. These questions were designed as single-choice questions with 5-likert scale. Besides, participants were also asked to openly provide feedback about their opinions on the interface and ideas to improve it. We sent survey invitations to over 300 users and 15 participants completed the survey. In the following section, we will present the results of user study.

5.1. Effects on the Effectiveness in Managing Conflict

For each feature of ScaffdCF, participants were asked whether it can help to more effectively manage the conflict, compared to GitHub's existing peer review interface. The

answer options were designed as 5-likert scale (from "strongly agree" to "strongly disagree"). As Figure 3 shows, among 7 major features, 4 were reported positively (*i.e.*, the median score is above 3). However, "*Decomposition into sub-issues*" feature was reported negatively, that is most participants thought that this feature is useless for managing conflict (*i.e.*, the median score is lower than 3). Besides, two features, "*Explicitly Expressing Agreement*" and "*Context (Prior Experience)*", were reported neutrally (*i.e.*, the median score is 3).

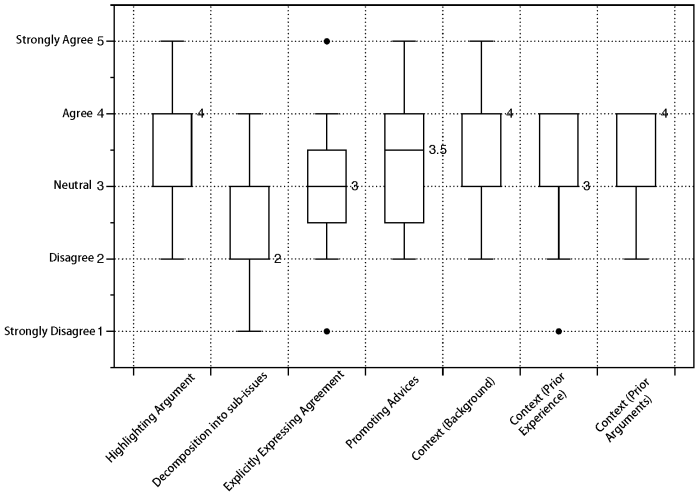

Figure 3. Survey Participants' Rating on the Effectiveness of ScaffdCF in Managing Conflict

"*Decomposition into sub-issues*" feature is not preferred, a possible reason is that in reality the sub-issues are highly interrelated, so it's not feasible to divide a pull-request into multiple sub-issues. And this feature might contribute little for conflict management, because some related information about the conflict on a particular sub-issue might be within other sub-issues' scopes and it's hard for reviewers to be aware of the whole situation. Participants also do not prefer the "*Promoting Advices*" feature. One potential reason is that the promotion of giving advices is not directly related to conflict management and participants do not realize the effectiveness of giving advices. For "*Context (Prior Experience)*" feature, a possible reason for its ineffectiveness is that only knowing about some numbers of a contributor or reviewer's prior work is not enough to judge his/her expertise or reputation.

5.2. Effects on Cognitive Overload

For the adoption and success of ScaffdCF, another important criteria is that ScaffdCF would not significantly increase the cognitive overload for reviewers to use. The cognitive overload might include the extra effort to understand the context information, maintain the reviews on multiple sub-issues, and figure out advices. For each feature of ScaffdCF, participants were asked whether it would increase the cognitive overload to use, compared to GitHub's existing peer review interface. The results (see Figure 4) shows that "*Decomposition*

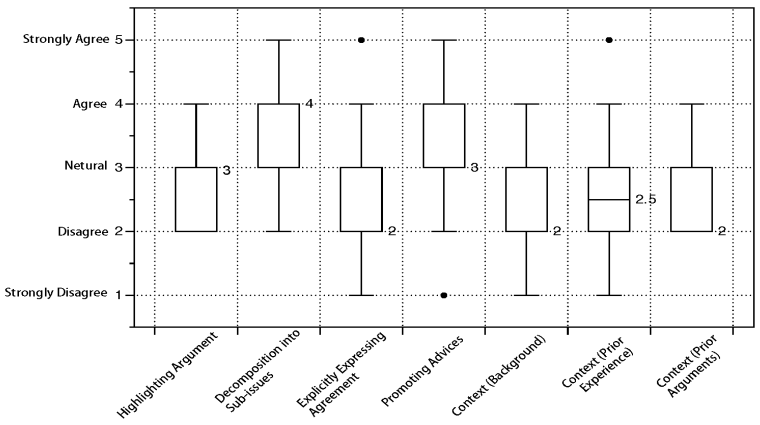

Figure 4. Survey Participants' Rating on the Cognitive Overload of using ScaffdCF

into sub-issues" feature significantly increase reviewers' cognitive overload. Participants' feedback to the open question suggested that this feature made reviewers pay more efforts to switch back and forth between multiple sub-issues. "*Promoting Advices*" and "*Highlightling Arguments*" were reported neutrally. The other features were reported negatively, that is these features would not significantly increase reviewers' cognitive overload.

6. Limitations and Improvements

In this section, we discuss some limitations and improvements about ScaffdCF. Although "*Decomposition into sub-issues*" is not favored by most participants, an improvement is to make this feature as an option setting. When a particular issue (significantly different from others' discussion) arose, the tab for this "sub-issue" can be created, but it should be supported as an option that the discussion about this "sub-issue" can be integrated into the general discussion flow, because some people might prefer to read the comments naturally as they flow.

Regarding to "*highlighting arguments*" feature, if the arguments were detected by automatic text classifiers, a potential risk is that a reviewer would feel being rudely treated when his/her comments were mistakenly classified as arguing comments. So it should be cautiously considered to use automatic text classifiers to detect arguments.

Another suggestion is about the context information. In the presenting interface, the contexts about prior experience and prior argument are just some number of one's prior activities. This is not informative enough. It should be better if the collaboration graph (collaboration on a project, pull-request, or reviewing work) of contributor and reviewers can be given.

Besides, although we tried to reach survey responders as much as possible, the sample size of this study is still relatively small (15 participants). We hope future research can conduct more in-depth evaluation on this prototype interface with more participants.

7. Conclusion

This paper designed a prototype interface, named "ScaffdCF", which provides various scaffolding features for project reviewers or administrators to manage conflict more effectively. We identify three problems of existing peer review systems and designed new features to scaffold conflict management process. The survey-based user study shows that most features can help to manage conflict more effectively without significantly increasing administrators' cognitive overload.

Acknowledgments

This work is supported by National Natural Science Foundation of China (NSFC) under grants No.61272533, No.61332008 and No.61233016.

References

[1] R. R. Blake, J. S. Mouton and A. C. Bidwell, "Managerial grid", Advanced Management - Office Executive, vol. 1, no. 9, **(1962)**, pp. 12-15.
[2] R. A. Cosier and G. L. Rose, "Cognitive conflict and goal conflict effects on task performance", Organizational Behavior and Human Performance, vol. 19, no. 2, **(1977)**, pp. 378-391.
[3] M. Deutsch, "Conflicts: Productive and destructive*", Journal of Social Issues, vol. 25, no. 1, **(1969)**, pp. 7-42.
[4] T. Erickson and W. A. Kellogg, "Social translucence: an approach to designing systems that support social processes", ACM transactions on computer-human interaction (TOCHI), vol. 7, no. 1, **(2000)**, pp. 59-83.
[5] A. Filippova and H. Cho, "Mudslinging and Manners: Unpacking Conflict in Free and Open Source Software", In Proc. CSCW '15, **(2015)**, pp. 1393-1403.
[6] I. U. Haq, "The impact of interpersonal conflict on job outcomes: Mediating role of perception of organizational politics", Procedia-Social and Behavioral Sciences, vol. 25, **(2011)**, pp. 287-310.
[7] P. J. Hinds and D. E. Bailey, "Out of sight, out of sync: Understanding conflict in distributed teams", Organization Science, vol. 14, no. 6, **(2003)**, pp. 615-632.
[8] K. A. Jehn, "A qualitative analysis of conflict types and dimensions in organizational groups", Administrative Science Quarterly, **(1997)**, pp. 530-557.
[9] A. Kittur, B. Suh, B. A. Pendleton and E. H. Chi, "He Says, She Says: Conflict and Coordination in Wikipedia", In Proceedings of CHI '07, **(2007)**, pp. 453-462.
[10] M. Lippi and P. Torroni, "Context-Independent Claim Detection for Argument Mining", **(2015)**.
[11] E. K. Lua, R. Chen and Z. Cai, "Social Trust and Reputation in Online Social Networks", **(2011)**.
[12] R. Padhye, S. Mani and V. S. Sinha, "A Study of External Community Contribution to Open-source Projects on GitHub", In Proceedings of MSR 2014, **(2014)**.
[13] R. M. Palau and M. F. Moens, "Argumentation mining: the detection, classification and structure of arguments in text", **(2009)**.
[14] A. Rahim and T. V. Bonoma, "Managing organizational conflict: A model for diagnosis and intervention", Psychological Reports, vol. 44, no. 3c, **(1979)**, pp. 1323-1344.
[15] Stuart H.C., Dabbish L., Kiesler S., Kinnaird P. and Kang R. Social Transparency in Networked Information Exchange: A Theoretical Framework. In Proc. CSCW '12 **(2012)**.
[16] Stvilia B., Twidale M.B., Smith L.C. and Gasser L. Information quality work organization in Wikipedia. Journal of the American Society for Information Science and Technology 59, 6 **(2008)**, 983-1001.
[17] Thomas K.W. Conflict and conflict management: Reflections and update. Journal of Organizational Behavior 13, 3 **(1992)**, 265-274.
[18] van Wendel De Joode R. Managing conflicts in open source communities. Electronic Markets 14, 2 **(2004),** 104-113.
[19] Wenjian Huang T.L.H.Z. Effectiveness of Conflict Management Strategies in Peer Review Process of Online Collaboration Projects. In Proc. CSCW'16 **(2016)**.

Authors

Wenjian Huang, is a PhD student in School of Computer Science, Fudan University. His research interests include Computer Supported Cooperative Work (CSCW) and Social Computing.

Tun Lu, is an associate professor at the School of Computer Science, Fudan University, China. His current research interests include CSCW, collaborative and social computing and HCI.

Ning Gu, is a full time professor in School of Computer Science at Fudan University. His research interests include Computer Supported Cooperative Work (CSCW), Cooperative and Social Computing, Human Computer Interaction (HCI).

