Mudslinging and Manners: Unpacking Conflict in Free and Open Source Software

Anna Filippova, Hichang Cho
Communications and New Media
National University of Singapore, Singapore
{annaf, hichang_cho}@nus.edu.sg

ABSTRACT
As the nature of virtual work changes, so must our understanding of important processes such as conflict. The present study examines conflict in ongoing virtual teams by situating itself in the context of Free and Open Source Software (FOSS) development. A series of semi-structured interviews with diverse representatives of the FOSS community highlight differences in the way conflict occurs. Specifically, a transformation of conflict types is observed together with a form of conflict previously unidentified in work on virtual teams. Findings suggest that the changing structure of ongoing virtual teams has important consequences for team processes like conflict.

Author Keywords
Virtual Teams; Free and Open Source Software; Conflict; Ongoing teams; Peer-Production; Computer-Supported Work; Computer-Mediated Communication.

ACM Classification Keywords
K.4.3 [Computers and Society]: Organizational Impacts---Computer-Supported Collaborative Work.

General Terms
Human Factors; Management.

INTRODUCTION

Virtual work is continuing to evolve, and so must our understanding of the foundational processes that underpin on-line collaboration. Teams are increasingly shifting away from fixed membership towards dynamic composition [37,38]. Team boundaries are becoming more fluid, while team members are more likely to be assigned to multiple teams [37]. At the same time, virtual teams are now increasingly less likely to experience life cycles with distinct start and end points [35]. Given that team age and structure significantly affect group processes and outcomes in virtual teams [21], these changes have important consequences for our understanding of virtual work.

Conflict is an important team process that has received a lot of attention from literature on temporary virtual teams: teams with fixed membership structures that experience short and finite life cycles [19,21,26]. However relatively less attention has been given to conflict in ongoing virtual teams: teams with dynamic membership composition that can progress for several years [11,14,35].

Free and Open Source Software (FOSS) development represents a unique opportunity to examine conflict in ongoing distributed teams in a natural setting. Due to the largely voluntary nature of participation, FOSS teams have fluid team boundaries, with team members frequently belonging to several projects [15]. Additionally, FOSS teams rarely have fixed end points, with notable projects continuing for many years [8].

Despite the potential for informing virtual team research, there is a lack of existing empirical work on conflict in FOSS [8]. The limited studies performed to date have been largely descriptive, focusing on a few case studies of well-established projects with strong free software ideals [12,13]. The present study therefore aims to build a more complete understanding of conflict in FOSS as an example of ongoing virtual teams. We do this by examining the process more generally in diverse representatives of FOSS communities with different ideologies, age and success rates.

While some important work has been done on conflict in Wikipedia [6,24,25], FOSS teams differ in several significant ways that may affect how conflict occurs. Aside from differences in the nature of output produced, FOSS teams can also be relatively more hierarchical than Wikipedia [33], with disagreements arising from authority issues [29]. In addition, while the Wikipedia community values maintaining a "Neutral Point of View" [27], conflict in FOSS teams is not only their defining feature [23], it is considered an art form [5].

Through an exploratory study of FOSS teams, this paper seeks to understand how the unique structure of ongoing teams affects team processes such as conflict. We propose the following research question:

CSCW '15, March 14–18, 2015, Vancouver, BC, Canada.

http://dx.doi.org/10.1145/2675133.2675254

RQ1: How do ongoing virtual teams such as FOSS development teams experience conflict?

Ongoing virtual teams must tackle process, structural and other socio-technical issues that evolve over time, whereas temporary virtual teams tend to focus primarily on the task at hand [35]. As such, conflict in ongoing virtual teams is more complex [14]. The anticipation of working together again may make it necessary for ongoing virtual teams to surface issues that temporary teams are able to suppress [35]. Additionally, unresolved conflicts may evolve and transform over time [34]. Yet these processes have not been well tested. To the best of our knowledge, only one study to-date has examined the transformation of conflict types in Wikipedia [1]. Crucially, the few initial studies focused on FOSS conflict, while not examining conflict sources directly, allude to disagreements arising from sources that do not fit neatly into the classical conflict taxonomy proposed by literature thus far [12,13]. We expand on these in the following section.

Taken together, the literature suggests that more research is needed to uncover how hybrid and new types of conflict manifest and transform in ongoing virtual teams like FOSS. However, most related findings are rather preliminary or based on observations of qualitatively different Wiki communities. Hence we aim to advance this line of research further by asking the following research questions:

RQ2: Do conflict types transform over time in ongoing virtual teams such as FOSS development teams?

RQ3: Do ongoing virtual teams such as FOSS development teams experience other sources of conflict beyond the traditional model?

The following section will briefly summarize the existing understanding of conflict in temporary virtual teams. We will also explore theory and related work that help to predict how conflict may occur differently in ongoing virtual teams such as FOSS development teams.

CONFLICT IN SHORT-TERM VIRTUAL TEAMS

Research on conflict in temporary virtual teams has typically separated this process into three distinct types: task, relationship and process conflict [19].

Task conflict in virtual work involves disagreements about the content of the task at hand, that is, about what needs to be done [20]. Task conflict arises out of a different interpretation of the team's goal or aim. Task conflict has the potential to generate a greater diversity of opinions and improve team function [21]. However, empirical research on virtual teams has found mixed results for the effectiveness of task conflict in improving team outcomes [18,28]. Task conflict has been found to frequently co-occur with other conflict types, in which case a reduction in its positive effects on team development is observed [3,11].

Relationship (or "affective") conflict involves interpersonal incompatibilities, and occasionally feelings of annoyance, frustration, and irritation with teammates [20]. Relationship conflict has been traditionally conceptualized as having negative effects on team function [19,28]. This effect is amplified when relationship conflict occurs together with process conflict [3,11].

Process conflict evolves out of differences regarding how the team's goal should be accomplished, and involves disagreements regarding resource and duty delegation [20]. Research in virtual work has found mixed support for the effects of process conflict on outcomes, because it overlaps conceptually with task conflict [3] and co-occurs with other conflict types [11].

CONFLICT IN ONGOING VIRTUAL TEAMS

Ongoing virtual teams have qualitatively distinct features from temporary teams, and therefore are likely to experience the process of conflict differently. Ongoing teams have a greater long-term orientation of working together in the future, a "shadow of the future" [35], and thus evolve unique intragroup processes [10]. For instance, trust operates differently in ongoing virtual teams [16]. Ongoing teams are also more likely to accumulate common knowledge to aid in more adaptive coordination [4].

While several studies examine ongoing virtual teams, they do not address in great depth the issue of long-term team dynamics, such as conflict. Furthermore, some studies conceptualize ongoing teams as either management teams [39] or simply teams having longer time frames for tasks with fixed end points assembled on an as-need basis [32]. This differs from the emerging teams without fixed end points and fluid, voluntary membership structures that we describe here. Below we elaborate on the two main differences we may find in conflict within ongoing virtual teams.

Transforming Conflict

The above findings on virtual teams suggest that different conflict types may occur at the same time and interdependently influence group outcomes [11]. In ongoing teams, conflict also has the potential to transform from one type into another [14]. Because ongoing virtual teams do not have fixed end points, they have more time and potential to evolve. Thus, if conflicts are not resolved initially, due to the longer time frames involved, conflicts may transform [34]. Additionally, the constant flux of members in ongoing teams who join at different times leads to different perspectives and interpretations of the same conflict. We expect to see not just co-occurrence but an evolution of conflict types in ongoing virtual teams. We refer to such hybrid conflict types as transforming conflict.

Rahim [34] has theorized two variations of transforming conflict: relationship conflict that may evolve into task conflict when team members indirectly express their

negative affect through disagreements about the task; and task conflict that can transform into relationship conflict, especially if the conflict is unresolved, or when frustrations are already running high in the group. Work on ongoing virtual teams has yet to examine these variations empirically.

One exception is recent work on Wikipedia that shows initial support for transformative conflict [1]. Arazy, Yeo and Nov [1] have found that task conflict can evolve into both relationship or process conflict, and that this transformation leads to a reduction in team performance. We need to extend this line of research further to better understand the different transitions conflict types can take. Do these findings hold for ongoing virtual teams such as FOSS teams? Do other kinds of transitions occur? When does one conflict type evolve into another?

Normative Conflict

Because ongoing teams have more time to evolve than temporary virtual teams, they are more likely to develop distinct group norms [35]. In fact, Ke and Zhang [22] found that FOSS teams develop their own community norms, and that these can vary significantly from project to project. Additionally, FOSS teams are a recursive public [23]: they are able to not only view and modify the source code to their software, but also reflect on and adapt their own practices and conditions of production.

As a result, ongoing teams such as FOSS development teams may experience normative conflict, that is disagreements about the norms of the group itself [30]. Normative conflict arises from a dissonance between the descriptive norms in a group, or observed behavior, and prescriptive norms, or expected group behavior [31]. In other words, when a group member observes the behavior of the group is no longer in line with team expectations, they may express dissent leading to conflict within the group.

Normative conflict is conceptually distinct from task, relationship and process conflicts because it is one level of abstraction higher. For example, rather than disagreeing about what task to perform, or how to perform the task of writing software, normative conflict may involve disagreements about project policy, direction, or ideology. This is a logical practice for ongoing teams that hope to survive for longer periods of time, because the group needs to be able to recursively adjust its own norms and procedures in response to critical events [35].

Several studies on FOSS teams, though not examining conflict sources directly, provide initial evidence that normative conflict may occur. For instance, Elliot and Scacchi [13] documented a disagreement that emerged in BOS because the descriptive norms of the community in using proprietary software were not in line with the prescriptive norm of using only Free Software. In a later study, Elliot and Scacchi [12] observed similar conflict patterns in the GNUe project as well. Both case studies involve well-established FOSS projects with a strong normative drive towards using only Free Software in all aspects of the projects' function. Though these studies provide initial hints of different conflict sources, we need to empirically examine conflict process in FOSS in greater detail among projects of different size, age and with ideological assumptions.

The next section will outline the methodology used in this study to explore how conflict manifests in FOSS teams as examples of ongoing virtual teams.

METHOD

To address our research aims, we conducted 16 semi-structured interviews with software developers who have contributed code to a FOSS project or released code under a FOSS license. Interviews were chosen as methodology for this exploratory study to allow for categories of conflict to emerge that would be given by, and meaningful to, the community being studied.

As the aim of this research was to explore conflict patterns among different types of FOSS projects, we placed emphasis on sampling a diverse set of participants. Developers were selected to represent a variety of positions in the community, level of experience, location and the type of problems their projects solve. Participants were invited to participate face-to-face at technology events and meet-ups, or by e-mail. Participants also recommended other members of the community, who were then selected for participation based on the same criteria, to avoid sampling homogenous clusters of developers.

We used GitHub profiles and filtering interview questions to assess levels of experience of our participants both in terms of length of time spent contributing to the project as well as volume of contributions made. Based on the hierarchical, or onion-like, model of FOSS organization [9], we also interviewed developers at different levels: committers, maintainers, core contributors, project founders (of both simple and complex projects) and community advisors.

4 women and 12 men participated in this project. Women currently make up about 11% of FOSS developers according to a recent survey [2]. Given the small number of participants in our study, for better gender representation, we oversampled and interviewed 4 women.

The final sample represented 7 countries: Australia, China, India, Malaysia, the Netherlands, Singapore, and the United States, and 14 projects. Projects included libraries: BitcoinJS, libXML, PrawnPDF; languages, tools and frameworks: Git, Grok, Node.JS, Python, Ruby on Rails, Zope; and end-user applications: Debian, Mozilla Firefox and Thunderbird, Plone, ScrollBack and Ubuntu. Though varying in size and stage of development, all the above projects are active and at least one year old, with a majority

spanning several years. They therefore represent examples of ongoing virtual teams that have had a chance to evolve.

To avoid priming interviewees about conflict, participants were asked to discuss their experiences in contributing to FOSS more generally, such as motivations, decision making in the team and memorable instances of disagreement. Additionally, participants were not limited to current conflict episodes - conflicts across different aspects of the project lifespan were discussed. Interviews lasted approximately one hour and were transcribed verbatim for further analysis. Participants were encouraged to share additional materials after the interviews, such as news articles. These materials supported the analysis process.

We applied a grounded approach to examine the common patterns of conflict in different FOSS projects [17]. First, all anonymised transcripts were open coded to identify conflict sources and their trajectories. In the second stage, we compared each instance to other examples across all the interview transcripts, identifying conceptually similar groups. We iterated this process until conceptual saturation, that is, until no new behaviors and inferences were being captured by the emerging categories.

RESULTS

This study explores how conflict occurs in ongoing virtual teams through the unique context of FOSS development. Our findings suggest that firstly, FOSS teams experience conflicts arising from constraints such as time zones and communication overload, as well as dependencies on other software projects. Second, due to their ongoing nature, FOSS teams experience transformation of conflict sources. Finally, due to their long-term orientation, FOSS teams experience conflict types previously unobserved by literature, such as normative conflict. These conflict sources are summarized in Table 1.

Socio-technical nature of conflict

RQ1 explores how conflict manifests in FOSS teams as examples of ongoing virtual teams. To address this, we begin by describing conflict sources that emerge from the specific conditions of work in FOSS. We identify structural limitations such as time zone and communication channels that affect conflict, as well as technical features of software production such as version clashes and dependencies on other software projects that give rise to conflicts about competing technologies.

Structural Sources

Though FOSS developers occasionally meet face-to-face to encourage community building, most important decisions are made on archived mailing lists or internet relay chat (IRC) meetings to allow the whole distributed team to participate [7]. In practice, this presents some challenges. Several participants have reported conflicts arising from communication issues such as different time zones, having too many communication sources to monitor, information overload, and difficulties being heard (P2, P8, P13, P4, P5, P12).

Several participants have noted that they have to stay up late at night in order to interact with team members in Europe and North America (P4, P5, P13). For many projects IRC, a synchronous communication tool, is used to co-ordinate important issues such as obtaining feedback on a proposed feature before making a formal pull request (P8) or to ask more experienced contributors questions (P4, P5). Some participants express frustration that developers from Asia and Australia need to conform to North American time zones (P4, P5, P13).

Developers also report conflicts arising from communication overload and problems being heard (P13, P8). For instance, particularly in larger projects, a steady flow of communication comes through the mailing lists every day as subsequent time zones come online:

"[You] can't subscribe to everything these days, [and] can't read everything subscribed. [By] the time you're done a whole bunch of other e-mails come in. [It's] a whole global thing […] people go to sleep, the next time zone wakes up and starts replying." (P13)

Additionally, projects may use multiple communication channels for decisions that are not always well publicized, resulting in an impression that decisions happen "behind closed doors" (P2). For P13, the inability to follow all possible communication channels has left the packager out of an important decision regarding their project:

"There was a decision that happened at some random IRC session that I didn't know about. [...] One day I just woke up [and] I read on some Ubuntu blog somewhere, "Banshee is no longer default". I'm like, "Huh?"" (P13)

P13 was unable to weigh in on the decision to select the default Ubuntu media player, thus another package was chosen instead. P13 raised the issue on IRC, linking notes from the missed meeting. Conflict erupted when other Banshee supporters expressed their solidarity by "spamming [flaming] the whole meeting notes upside down" (P13).

Some developers have reported issues being heard even when they submit pull requests (P8, P1):

"One thing I do disagree with is [people] who just don't respond. [...] Thank you for letting me use your tool for free, [but] I went to effort to umn contribute that thing back, umn it would be nice if you could at least just respond to say no. [I've] accidentally done that to a few people. And I feel like an asshole every time". (P8)

Because it is relatively easy to create an issue or give feedback on a platform like GitHub or Launchpad, developers find that many users open "dumb issues" and as a result their contributions "get lost in the noise" (P8).

Conflict Source	Conflict Type	Examples
Structural	Procedural and Relational	Time zone differences, difficulties keeping up with multiple communication channels, information overload, noise in communication channels
Competing Technologies	Task	Version conflicts due to backward-compatibility issues, multiple alternative libraries/software dependencies
Transforming	Task–Relationship	Disagreements with authority figures, frustrations due to lack of task progress
	Task–Process	Emergence of procedural errors during task conflict discussions
	Process–Relational	Disagreements among project leaders about important procedures that become personal, personal attribution of blame following procedural mistakes
Norms	Normative	Disagreements regarding project policies, governance issues, ideological clashes arising from community interaction rather than the task at hand.

Table 1. Summary of conflict sources and types

Finally, the amount of conflict experienced in a team varies with the level of interdependency in the project. P8 points out that Node.JS has a modular codebase, and therefore, "if you disagree with someone's choice, you can [just] replace the piece". "In Node, there is less need for people to all agree on something." (P8) On the other hand, the NPM project is more centralized in it's code architecture design, and:

[It] has all of the same problems as [you would] expect […] a whole bunch of people who have different uh concerns, different priorities and different needs out of a piece of software […] They're all like vying to get NPM to suit their needs. (P8)

Thus, because a project's technical structure is more interdependent, their social structure also experiences greater friction.

Taken together, the above findings highlight that just like temporary teams, larger ongoing teams experience difficulties with information overload [18]. Similarly, teams that rely on synchronous communication also suffer from issues of distance and time. Finally, the amount of friction in a team can vary with the level of code base interdependence. That is the social processes within a group change depending on the technical conditions

Competing technologies

A second source of conflicts that arise during FOSS development involve competing ideas about the appropriate technology to use to solve a problem, or competing versions of the same technology. A number of the surveyed developers reported such conflicts (P13, P10, P9, P8). For instance, in the Banshee example presented in the previous section, difficulties of communication have compounded an existing conflict between supporters of two competing technologies, Banshee and Rhythmbox:

"There was a lot of mudslinging [flaming] going around as well by people who wanted Rhythmbox as the default. [...] I think a lot of the perception was that, "Oh no, it's because Banshee uses MONO that's why it's slow" and all that. It's mostly the same stigma that happens with Java. Except with Java it's true." (P13)

Disagreements between supporters of different projects are common, and often turn into flames. This observation is consistent with previous work by Coleman [5] who highlighted that disagreements among FOSS developers about different technologies can occur for sport, that is, be an end in themselves.

However, our findings show that beyond their autotelic function, conflicts about competing technologies are also able to influence the social dynamics of a project. For example, P10 recalls having to negotiate different contributors' suggestions for libraries to include in the project:

"So this person jumped in, and be like, 'Hey, I have a library. Use mine!' And then the other person, who has commit rights, and he jumped in, he was like, 'I also have a library'" (P10)

Such disagreements occur at an intersection of competing priorities and agendas, as contributors may prefer to include their own libraries to enhance their reputation in the community or due to familiarity with the code base.

Because FOSS projects depend on the efforts of volunteers, the maintainer is challenged to weigh these suggestions against other approaches and come up with a solution that satisfies both the volunteers and projects' aim:

[...] and then there's also a big number library, which is written by someone else, which is more popular than both of their libraries combined, so I'm like, "How about, I don't know, let's use the standard?" (P10)

Negotiating conflicting opinions on project dependencies is difficult not only because choosing a side means rejecting the suggestion of some team members. Software dependencies also have their own teams behind them. Thus, choosing to include an outside project also means choosing to depend on the project's community for parts of your own project function.

Additionally, more mature projects sometimes find themselves in conflict between old and new versions of the same software. While version upgrades are commonplace, breaking compatibility with earlier versions is sometimes required in order to make substantial improvements to the project and "fix early mistakes" (P8). This can create faultlines in the community because rewriting old code to conform to the new version is too time consuming:

"Yeah that's all very nice but I have a lot of code written in Python 2 and what's gonna happen right? What's gonna happen, not just to this code, but to the community where some people will write like Python 3 code, some people will write Python 2 code" (P9)

These faultlines can generate community-wide friction and conflict between supporters of the opposing versions (P9).

Taken together, the above examples highlight that disagreements between developers about conflicting versions or competing software projects significantly impact community function. They not only advance technical decisions about the project, they also become critical junctures that influence the social structure of the team going forward.

Transformative conflict

The majority of conflicts reported above can be classified into conflict sources based on Jehn's taxonomy [20]. Task conflict is expected in any teams that have a certain degree of interdependence of task, and as we have seen above, FOSS teams frequently need to negotiate code commits as well as issues related to procedure and personal disagreements. However, more interestingly, the conflict episodes uncovered in this study have a tendency to evolve into different conflict types over time. Thus to address RQ2, the next section will present examples of such transformations, and discuss their significance with relation to the changing nature of ongoing teams.

Task-Relational conflict

Firstly, out findings highlight the transformation of task into relational conflicts (P9, P1, P8). Often, the relational conflicts described were the result of already mounting frustrations with a task, or involved a power imbalance between disagreeing parties.

For example, the lead of the Zope project approached one of his maintainers after a conference and chastised him personally instead of the team for not being more careful when making changes to the code base (P9). P9 suggested that this task conflict turned personal because of already mounting frustrations during the conference about the lack of progress in the project in general:

"And in part it was a reaction to my negative sort of attitude, [and] all this frustration behind me already to get this small progress. [...] So we had this sort of discussion and we had been friends [for] years by then [and] that was the last time actually I met him in real life". (P9)

Interestingly, participants often reported relational conflicts emerging from task conflicts with authority figures (P9, P1). P1 suggests this may be because project maintainers and leads have greater demands on their time, and are also in a position of authority that allows them to get away with being more offensive.

One respondent suggested that this transformation into relationship conflict could be avoided if contributors choose not to make the issue personal (P1). Some have also suggested that it is not worth arguing with a maintainer even if they may be right because the maintainers have control over what features may be accepted:

"You want to keep those people who are in charge of okaying or not okaying, keep them on your side. But if you [...] throw a tantrum when they don't accept your feature, they are less likely to take you seriously" (P8).

Taken together these findings indicate that in FOSS teams, task conflicts can evolve a relational dimension over time due to the ongoing nature of the teams, as frustrations about a lack of progress build up over time. Interestingly, task conflicts can also evolve into relational conflicts due to power imbalance and harsh communication style of some authority figures. Given the long-term focus of the projects, developers recognize that it is better for the overall health of the team and their relationships with authority figures to accommodate the individuals in charge.

Task-Process conflict

Some FOSS projects have a 'way' of doing things within the group that may be unwritten, such as a style of writing code (P10, P2, P1) or accepted practices for division of labor (P8, P2, P9, P3). Disagreements about a task can evolve into procedural conflicts when developers are not aware of this style, or have a strong opinion about doing things differently.

For instance, P1 recalls another contributor having very poor success getting his contributions into the Git project due to disagreements on the style the code should be written in:

"What he is doing, adding a new feature, [is] something that Git cannot do, it's just the way [...] the code is doing it, they prefer it to be done in another manner but he doesn't agree" (P1)

In the above example, discussion about process took over the discussion about task because the contributor refused to adjust to the project's coding style.

In another example, a maintainer for the Zope project sparked a conflict when he removed a list of dependencies without discussing the decision with his teammates (P9). The other maintainers disagreed with his decision to remove the dependency list, resulting in a task conflict. However, when it emerged that procedure wasn't followed, and this change wasn't discussed with other members who may still depend on this list, the conflict became procedural. Interestingly, when P9 attempted to revert this change, in essence fixing the task element of the conflict, he also faced backlash from the community for going against procedure.

Taken together, the above examples highlight a prioritization of conflict occurring in ongoing virtual teams. When disagreements about the task at hand begin to overlap with conflict on how the task should be performed, procedural issues appear to be given more weight than task related ones. This may be because procedural issues are relatively more important for ongoing teams to address and resolve to improve coordination and ensure project longevity [29]. Thus some projects would rather reject a significant code contribution if it does not fit in with the project coding style, and suspend activities on task related disagreements until a procedural issue is resolved.

Process-Relational conflict

Our findings have also uncovered a combination of conflict types theorized by previous literature [34], but not yet well-documented by research on ongoing virtual teams [1]. Specifically, participants have reported a number of examples of procedural issues that evolve into relational conflict.

For example, following a prominent incident in which the Linux kernel was hacked, a disagreement emerged between two leaders, Linus Torvalds, and Shawn Pierce, about implementing tighter contribution security in their software version control system. (P1). Though the task at hand to improve security was clear, the leaders disagreed on the procedure to accomplish this. The conflict escalated to a relational one, when Shawn suggested an alternative solution to the one Linus proposed, and

"Linus said 'No, totally wrong'. Except he didn't use those words, he used something worse". (P1)

According to P1, Linus Torvalds' harsh tone had transformed a procedural issue into a relational one, just like in some of the task-relational conflicts above. Though Linus is known for his specific communication style, the previous sections have documented some other examples of project leaders exhibiting similar patterns due to their positions of authority. This suggests that in projects with more pronounced hierarchies, there is a greater tendency for conflicts to take on a relational dimension when they involve project leaders.

We also observe situations in which process conflict can become relational among regular committers and maintainers.

For instance, when Debian developers with commit access wish to modify code they are not themselves responsible for, they are expected to follow an unwritten rule (P13). Committers need to submit their Non-Maintainer Upload with a delay (7 days) to allow the maintainer of this code to review and accept the changes. A Debian maintainer points out that other maintainers get "very angry" if code is uploaded without adhering to this convention (P13).

Debian has evolved the above community norm to avoid procedural conflict transforming into relational conflict. This once again highlights the long-term focus of ongoing virtual teams, who have time to recursively adopt social norms that help them prevent disagreements experienced in the past.

In general, the above section has shown that ongoing virtual teams such as FOSS development teams exhibit strong tendencies for conflict types to transform and evolve over time. In addition, more combinations of conflict transformations have been observed than those documented in previous work [1]. In particular, conflicts involving project leaders or an imbalance of power appear to frequently take on a relational dimension. At the same time, procedural conflicts appear to be given higher priority when they emerge from other conflict types. This suggests that while ongoing virtual teams evolve hierarchies and procedures to help them manage their community growth [29], this introduces new tensions in the project. However, the long-term orientation of on-going virtual teams allows them to iteratively adjust their behavior when moving forward.

Normative conflict

The previous section dealt with conflict that emerged due to misunderstandings about the coding task to be performed, project styles and procedures or interpersonal friction. Addressing RQ3, this section deals with conflicts that do not fit into this classification. We refer to them here as normative conflicts, as they involve a dissonance between the stated aims of the group, or prescriptive norms, and its practices, or descriptive norms [30]. Normative conflicts are thus distinct from procedural conflicts and issues of authority, because they occur at a level of abstraction

higher. Instead of disagreeing about how a task should be performed, or by whom, normative conflict involves disagreements about issues like project policies, governance structures, and project ideology.

For instance, following an incident where a few users were banned after being "quite abusive" (P13), Debian developers debated the practice of banning users on the project mailing list:

"There was a bit of a debate about [the] trigger happy banning behavior of some people. Some people were concerned that freedom of speech was being obstructed" (P13)

The Debian community was not only debating its own policies with respect to banning of users when referring to the obstruction of free speech. Developers were also concerned that the community was no longer living up to their own beliefs. The descriptive norms of banning individuals were thus misaligned with prescriptive norms of free speech in the community, causing a conflict.

Similarly, in a high profile incident within the Ruby on Rails project, a community member resorted to hacking the project to demonstrate a dissonance between the community norms and practices. The Rails community has strong opinions on how the programming framework should work and be used (P2, P8):

"Rails is opinionated software. It makes the assumption that there is the 'best' way to do things, and it's designed to encourage that way." [40].

The "Rails way" is the group's ideological construct that helps to align the efforts of contributors towards the same goal. However, this belief of the 'best way to do things' came into conflict with actual group practices when a developer uncovered a critical security flaw in Rails, but received little support in addressing the issue from the core team:

"He [Homakov] was pretty much ignored and put down because the core team thought it was a non-issue and that things just weren't done that way because it's not the Rails way" (P2)

To illustrate the severity of the security flaw, Homakov exploited the vulnerability to make unauthorized commits to the Rails project on GitHub. He also spoofed comments by prominent members of the Rails team, such as dhh, the project lead.

"Perhaps if he didn't feel that his point was worth proving we would be dealing with an insecure by default framework just because the core maintainers or a majority of them don't think that's 'the way' to do so". (P2)

Through this hack, Homakov pointed out that the prescriptive norms of Rails as the "best way" to write software were at odds with their actual practices because they overlooked the severity of a security flaw in the framework. By spoofing comments of prominent team members and essentially speaking for them, Homakov was also drawing attention to the fact that his attempts at improving the project were ignored.

Though Homakov was initially banned from GitHub for this incident, his account was reinstated shortly thereafter and the security flaw was fixed in the next version of Rails (P2), showing an indirect acknowledgement that Homakov made an improvement to the "Rails way".

Normative conflicts therefore have powerful implications for the team as a whole, as when they erupt they force the group to iteratively evaluate their collective goals and purpose, and if necessary, make changes so that group practices continue to be in line with their aims.

To conclude this section of the paper, we will present one final example illustrating that normative conflict is not immutable and can interact with other conflict types. The following conflict episode was independently discussed by a number of participants (P9, P1, P2, P8, P13, P16), and simultaneously involves procedural, relational and normative dimensions.

In late 2013, a developer submitted a pull request to libuv, a sub-project of Node.JS, replacing a gendered pronoun inside a line of documentation with a gender-neutral alternative (P9). Ben Noordhuis, the project maintainer, rejected the pull request for being "too trivial to bother" (P8). While for Ben the source of the disagreement was procedural, an issue of time management, the community saw this as sexist, a personal issue, and "went a bit crazy at him about it" (P8).

The conflict took on an additional procedural dimension when Isaac Schlueter stepped in. Isaac created one of the other foundational Node.JS programs, NPM, and has since left the project but maintained his upload access. Isaac accepted the pull request without any discussion, thus undermining Ben's authority as maintainer (P9). Ben, in turn, reverted Isaac's commit (P8):

"He [Ben] didn't revert it because he disagreed with it, he reverted it because [they] have procedures for getting stuff into libuv. And although Isaac has commit rights, he shouldn't [...] get around the process." (P8)

Isaac, on the other hand, argued that he was upholding the norms of the project, to be "inclusive of non-male people and that our language should reflect this explicit inclusion" [36]. The conflict gained a normative dimension in highlighting this disconnect between upholding the prescriptive norms of inclusivity and the way the norms were put into practice through documentation language.

As the above example demonstrates, it is possible for localized conflicts, such as about group processes, to bring to light higher order issues in the project that cause fundamental faultlines in the group. In the case of Node.JS, this conflict allowed the community to clarify several

community norms, such as inclusivity and when it is appropriate to revert others' commits.

Taken together, the above normative conflict examples highlight its power in driving forward team development in ongoing virtual teams that recursively evaluate and adjust their techno-social structures. Normative conflict therefore has potential to be generative to community health and function.

DISCUSSION

The present study aimed to explore the relatively understudied occurrence of conflict in Free and Open Source Software (FOSS) development teams as a special example of ongoing virtual teams. Ongoing virtual teams are characterized by changes such as more fluid membership structures and no fixed end points [35], which we expected to lead to changes in the experience of social processes like conflict.

Through an exploratory study of FOSS development teams we aimed to uncover how the process of conflict manifests (RQ1), transforms (RQ2) and differs (RQ3) in ongoing virtual teams. Our findings contribute several important insights.

Firstly, we find that conflicts can arise directly out of the technical constraints and features of FOSS development teams, such as communicating across time zones via multiple communication channels. This is consistent with literature on temporary virtual teams [18] and is likely to vary with team size. An interesting question is whether large Wiki communities experience similar frictions. This may not be the case because FOSS teams have a greater expectation for and reliance on synchronous communication through channels, like IRC.

Additionally, we find that conflicts can arise from incompatible software versions, and create faultlines in the social fabric of the group. Similarly, because FOSS teams produce software that frequently depends in some part on other FOSS projects, they are also likely to experience conflicts about competing technologies. Thus because FOSS teams are socio-technical systems, decisions they make about the technologies they build and borrow affect their social processes, like conflict.

Second, our findings highlight that conflict types occurring in ongoing teams frequently evolve. Consistent with earlier work on Wikipedia [1], this study observes task conflicts transforming into relational and process conflicts. Our study further highlights that process conflicts may also evolve into relational conflicts, suggesting that the transformations we observe may not be limited to only the types previously identified.

We find that issues of authority or imbalance of power appear to be significant factors contributing to this transformation. When conflicts evolve, projects prioritize addressing certain types of conflict over others. Ongoing virtual teams try to focus their attention on conflict types that are more significant to the long-term viability of the group. Some projects are even willing to reject a significant code contribution if it is not in line with the norms or processes of the group.

It would be interesting for future research to explore what other factors contribute to each of these transitions, and whether the way transitions are handled has an impact on team outcomes. Our findings also have implications for research interested in examining outcomes in ongoing virtual teams. Our results suggest that research designs should be carefully crafted to ensure that when a conflict episode is studied, it is measured in its entirety in relation to team outcomes, because viewing the conflict in cross section at different times may lead to inconsistent and possibly contradictory findings.

Finally, in addition to task, relationship, process and transformative conflicts, our results show that ongoing teams also experience normative conflict. Normative conflict deals with distinctly different kinds of intra-group disagreements, and involves a dissonance between the stated aims of a group and its practices [30]. In our findings we have seen normative conflict arising when the norms of free speech clashed with the practice of banning users, as well as when the norms of providing the "best way" to write software conflicted with an oversight of a major security flaw. Normative conflict can also evolve from other conflict types: when a procedural issue about contributing documentation occurred, it highlighted how existing practices were not reflecting the community aims to be gender inclusive.

We argue it is natural to find normative conflict in ongoing teams such as FOSS teams because they are recursive publics [23]. Recursive publics modify their own social and technical conditions, and thus they are not only able to reflect and adjust their technological foundations, but they also reflexively adjust their social foundations. Normative conflict is a channel that drives this process forward, and if handled well, allows ongoing virtual teams greater long-term sustainability.

To the best of our knowledge, ours is the first study to examine normative conflict in virtual teams. Work on normative conflict in collocated groups has only featured a handful of experimental studies thus far [31]. Thus this is a fruitful avenue for future research into ongoing teams, in particular in examining what effects normative conflict may have on group outcomes like performance, satisfaction and member attrition in a team.

Limitations and future work

The present study is not without limitations. Though all efforts were made to select participants such that there were no homogenous clusters of developers, the size and nature of the sample strategy may limit generalizability of the above findings. It would be interesting for future research to

examine these patterns in a broader survey of the FOSS community. Additionally, self-reporting conflict episodes may have a dimension of social desirability – other research methods that do not rely on self-reports, such as a textual analysis of richly available mailing list archives, could serve to further validate these findings.

Our study focuses on the intragroup level of analysis. There may be different patterns of conflict at the intergroup level. In particular, since FOSS projects have complex interdependencies between communities this may introduce additional complexity into their social processes. This is an interesting question for further research – how do different teams interact and coordinate with one another given the differences they may have in norms and ideology?

Finally, this paper does not look closely at the way organizational sponsorship of FOSS projects might affect their social dynamics, such as conflict. This is another potential avenue for future research to explore, as we know that a significant proportion of FOSS developers are paid to contribute to projects [15].

CONCLUSION

The present study set out to examine the way conflict occurs in ongoing virtual teams. We found that conflict types can evolve over time, while group norms recursively become a source of conflict for ongoing teams. At the same time, due to their long-term orientation, ongoing teams prioritize certain conflicts over others to ensure their resolution and sustainability. Our results inform research on virtual work, and suggest further research is needed to explore the way ongoing teams are changing our understanding of virtual team processes.

ACKNOWLEDGMENTS

We gratefully acknowledge our anonymous reviewers, whose insightful comments have helped to make significant improvements to this manuscript.

REFERENCES

1. Arazy, O., Yeo, L., and Nov, O. Stay on the Wikipedia task: When task-related disagreements slip into personal and procedural conflicts. *JASIST 64*, 8 (2013), 1634–1648.
2. Arjona-Reina, L., Robles, G., and Dueñas, S., *The FLOSS2013 Free/Libre/Open Source Survey*. (2014)
3. Behfar, K.J., Mannix, E.A., Peterson, R.S., and Trochim, W.M. Conflict in Small Groups: The Meaning and Consequences of Process Conflict. *Small Group Research 42*, 2 (2011), 127–176.
4. Caya, O., Bassellier, G., and Pinsonneault, A. Virtual Team Common Knowledge: Construct Specification and Effect on Knowledge Integration Effectiveness. *ICIS 2008 Proc*., (2008), 14.
5. Coleman, E.G. *Coding Freedom: The ethics and aesthetics of hacking*. Princeton University Press, (2013).
6. Collier, B. and Bear, J. Conflict, criticism, or confidence: an empirical examination of the gender gap in Wikipedia contributions. *CSCW'12*, ACM (2011), 383–392.
7. Crowston, K., Howison, J., Masango, C., and Eseryel, U.Y. The role of face-to-face meetings in technology-supported self-organizing distributed teams. *IEEE Transactions on Professional Communication 50*, 3 (2007), 185–203.
8. Crowston, K., Wei, K., Howison, J., and Wiggins, A. Free/Libre Open-Source Software Development: What we know and what we do not know. *Computing Surveys 44*, 2 (2012), 1–35.
9. Crowston, K., Wei, K., Li, Q., and Howison, J. Core and Periphery in Free/Libre and Open Source Software Team Communications. *Institute for Software Research.* (2006) Paper 489.
10. de Jong, B.A. and Elfring, T. How does trust affect the performance of ongoing teams? The mediating role of reflexivity, monitoring, and effort. *Academy of Management Journal 53*, 3 (2010), 535–549.
11. de Wit, F.R.C., Greer, L.L., and Jehn, K.A. The paradox of intragroup conflict: A meta-analysis. *Journal of Applied Psychology 97*, 2 (2012), 360–390.
12. Elliott, M. and Scacchi, W. Communicating and mitigating conflict in open source software development projects. *Projects & Profits,* (2002), 25–41.
13. Elliott, M.S. and Scacchi, W. Free Software Development: cooperation and conflict in a virtual organizational culture. *Free/Open Source Software Development,* IGI Global, (2005), 152–173.
14. Filippova, A and Cho, H. Below boiling point: negotiating conflict thresholds in distributed work. *IR'15 Proc.*, AoIR (2014).
15. Ghosh, R.A., Glott, R., Krieger, B., and Robles, G. Free/Libre and Open Source Software: Survey and study, *Working Paper,* International Institute of Infonomics, University of Maastricht (2002).
16. Gibson, C.B. and Manuel, J.A. Building trust. *Virtual teams that work*, (2003), 59–86.
17. Glaser, B.G. and Strauss, A.L. *The discovery of grounded theory: Strategies for qualitative research*. Aldine Transaction (2009).
18. Griffith, T.L., Mannix, E.A., and Neale, M.A. Conflict and Virtual Teams. In E.M. Turner, ed., *Groups at Work: Theory and Research*. Lawrence Erlbaum, Mahwah, NJ, (2001), 445–470.

19. Hinds, P.J. and Bailey, D.E. Out of sight, out of sync: Understanding conflict in distributed teams. *Organization Science 14*, 6 (2003), 615–632.
20. Jehn, K.A. and Mannix, E.A. The dynamic nature of conflict: A longitudinal study of intragroup conflict and group performance. *The Academy of Management Journal,* (2001), 238–251.
21. Kankanhalli, A., Tan, B.C.Y., and Wei, K.-K. Conflict and Performance in Global Virtual Teams. *JMIS 23*, 3 (2006), 237–274.
22. Ke, W. and Zhang, P. Motivation, Social Identity and Ideology Conviction in OSS Communities: The Mediating Role of Effort Intensity and Goal Commitment, *International Journal of Electronic Commerce 13,* 4 (2007), 39–66.
23. Kelty, C.M. *Two Bits: The Cultural Significance of Free Software.* Duke University Press, (2008).
24. Kittur, A. and Kraut, R.E. Beyond Wikipedia: coordination and conflict in online production groups. *CSCW'10*, ACM (2009), 215–224.
25. Kittur, A., Suh, B., Pendleton, B.A., and Chi, E.H. He says, she says: conflict and coordination in Wikipedia, *CHI 2007,* ACM (2007), 453–462.
26. Martinez-Moreno, E., Gonzalez-Navarro, P., Zornoza, A., and Ripoll, P. Relationship, task and process conflicts on team performance: The moderating role of communication media. *International Journal of Conflict Management 20*, 3 (2009), 251–268.
27. Matei, S.A. and Dobrescu, C. Wikipedia's 'Neutral Point of View': Settling Conflict through Ambiguity. *Information Society 27*, 1 (2011), 40–51.
28. Mortensen, M. and Hinds, P.J. Conflict and shared identity in geographically distributed teams. *IJCMA 12*, 3 (2001), 212–238.
29. O'Mahony, S. and Ferraro, F. The emergence of governance in an open source community. *Academy of Management Journal* 50, 5 (2007), 1079–1106.
30. Packer, D.J. On being both with us and against us: a normative conflict model of dissent in social groups. *Personality and Social Psychology Review 12*, 1 (2008), 50–72.
31. Packer, D.J. and Chasteen, A.L. Loyal deviance: testing the normative conflict model of dissent in social groups. *Personality and Social Psychology Bulletin 36*, 1 (2010), 5–18.
32. Powell, A., Piccoli, G., and Ives, B. Virtual Teams: A Review of Current Literature and Directions for Future Research. *SIGMIS Database 35*, 1 (2004), 6–36.
33. Prasarnphanich, P. and Wagner, C. The role of Wiki technology and altruism in collaborative knowledge creation. *JCIS 49*, 4 (2009), 33–41.
34. Rahim, M.A. *Managing Conflict in Organizations*. Transaction Publishers, New Brunswick, NJ, (2011).
35. Saunders, C.S. and Ahuja, M.K. Are All Distributed Teams the Same? Differentiating Between Temporary and Ongoing Distributed Teams. *Small Group Research 37*, 6 (2006), 662–700.
36. Schlueter, I., ed. *Comment on "The Next Phase of Node.JS."* [Retrieved 1/5/2014 from https://news.ycombinator.com/item?id=7064470]
37. Tannenbaum, S.I., Mathieu, J.E., Salas, E., and Cohen, D. Teams Are Changing: Are Research and Practice Evolving Fast Enough? *Industrial and Organizational Psychology-Perspectives on Science and Practice 5*, 1 (2012), 2–24.
38. Wageman, R., Gardner, H., and Mortensen, M. The changing ecology of teams: New directions for teams research. *Journal of Organizational Behavior 33*, 3 (2012), 301–315.
39. Zack, M.H. and McKenney, J.L. Social context and interaction in ongoing computer-supported management groups. *Organization Science 6*, 4 (1995), 394–422.
40. Anonymous. *Rails Guides* [Retrieved 25/7/2014 from http://guides.rubyonrails.org/getting_started.html]