Design, Discussion, and Dissent in Open Bug Reports

Andrew J. Ko and Parmit K. Chilana
The Information School | DUB Group
University of Washington
{ajko, pchilana}@uw.edu

ABSTRACT

While studies have considered computer-mediated decision-making in several domains, few have considered the unique challenges posed in software design. To address this gap, a qualitative study of 100 contentious open source bug reports was performed. The results suggest that the immeasurability of many software qualities and conflicts between achieving original design intent and serving changing user needs led to a high reliance on anecdote, speculation, and generalization. The visual presentation of threaded discussions aggravated these problems making it difficult to view design proposals and comparative critiques. The results raise several new questions about the interaction between authority and evidence in online design discussions.

Categories and Subject Descriptors

H.5.3 [Group and Organization Interfaces]

General Terms

Documentation, Design, Human Factors.

Keywords

Change requests, design rationale, open source, Bugzilla

1. INTRODUCTION

Millions of people benefit from open source software (OSS), and yet the extent to which open source software satisfies peoples' needs depends partly on the the design decisions made by small teams of distributed developers. It is surprising then that we know so little about how these design decisions are made in practice, apart from general studies of computer-mediated discourse [14,24]. For example, how do developers cope with the fact that software is generally formless, and thus difficult to discuss and describe in conventional computer-mediated tools? Since software is often designed to serve multiple distinct tasks, how do teams reach consensus in the face of competing concerns, and in many cases, anonymity? Furthermore, since small changes to software can have large effects on its behavior, how do software teams assess and evaluate the impact of a change?

To begin to answer these questions, we performed a detailed qualitative analysis of design discussions in *bug reports* from the Firefox, Linux kernel, and Facebook API projects. Bug reports in these projects represent concrete, actionable issues with a software project and are open to anyone who wants to help reproduce and address the problem or influence how it is dealt with. Although bug reports are not the only place that developers discuss design, they are one of the few places where design decisions are translated directly into code. This makes them a compelling place to uncover the structure of software design debate, understand how contributors defend opinions, and assess how opinions influence decisions. Furthermore, in addition to limiting our analysis to bug reports, we focused specifically on changes to existing software (as opposed to initial design) and on contentious (rather than routine) discussions, to help amplify the strengths and weaknesses of computer-mediated discussion tools.

Our results suggest that while OSS design discussions exhibit challenges similar to other domains [26,17] (understanding the design space, exploring alternatives, making tradeoffs, etc.), they also exhibit key differences:

- Discussions exhibited an underlying philosophical divide between achieving the original intent of a design or adapting to user needs. This divide did not fall along user/developer roles as many developers were quite user-centered in their arguments. This fundamental power dynamic calls for new forms of process scaffolding to support software design decision-making.
- The measurability of software qualities being discussed influenced whether consensus was achievable and to what extent commenters relied on anecdote, speculation, and generalization to support their claims. For example, discussions of performance were driven by testing and converged quickly, whereas qualities that required subjective or empirical interpretation (such as learnability, flexibility, and security) diverged and were ultimately decided by authorities instead of the community.
- The above challenges were aggravated by the unsuitability of online textual discussions for discussing changes to software behavior. Design proposals were lost among a sea of critiques, leading to redundancy and reiteration. Design critiques were detached from the ideas themselves, making it difficult to see on what evidence, if any, consensus was based. Also, because of the sequential nature of comments, the more a discussion involved user feedback, the harder it was to find it.

Before describing these results and their implications in detail, we first describe prior work and our methodology. We then discuss the implications of our results for distributed software teams.

2. RELATED WORK

There is considerable prior work on software engineering practice [1,4,13,15], computer-mediated discourse [14,24], and discussion of various forms [10,17,20], but little work focusing on discussion in distributed software teams. Recent work on a corpus of Mozilla bug reports briefly considered contentious reports, finding that they were rife with misunderstandings about the intent behind bug report discussion [16]. Twidale and Nichols describe a study similar to ours, finding a lack of usability expertise and difficulties in describing user interaction with text, but limited their investigations to usability discussions and focused on an ad hoc, exploratory sample of reports [25]. An interview study with 25 software designers found that developers are rational when choosing one of

iConference 2011, February 8-11, 2011, Seattle, WA, USA

several options but less so when iterating on a single option [27]. Another related study investigated "decision episodes" in emails and forums of six OSS projects, finding that most changes could be done with one author, but decisions about impact beyond code involved multiple contributors [20]. There is also considerable work on contexts in which software design discussions occur. For example, a study of information needs in collocated software teams [15] showed that software developers regularly use instant messaging, e-mail, and informal face to face conversation in order to make design decisions with their team. Developers also discuss design decisions in mailing lists [3], during code inspections [23], and sites designed to facilitate collaboration between developers and designers [2].

Studies of other aspects of bug reports have revealed findings about design discussion. For example, in a study of software developers' opinions about what makes "good" bug reports, Bettenburg et al. [4] found that developers give preferential attention to known reporters and that violating "netiquette," such as opening rude or sarcastic bugs, can influence the degree to which the bug gets fixed. A study of bug reports found that the bug reports of collocated teams contain only a small fraction of the key events a bug report's history and that team culture affects bug related activities [1]. There is evidence, for example, that distance increases bug fixing time because of increased communication requirements [9].

Outside of software design, there have been numerous studies of design decision making, both computer-mediated and face to face. For example, studies have also shown that computer-mediated discussions can be more focused [24], but take more time, involve more conflict, and have more difficultly leading to consensus [14,24]. Other studies show that informal social interactions and the awareness that arises from them, are crucial in overcoming attribution errors (blaming a collaborator, rather than the collaborator's situation) [8]. Studies comparing online discussion boards to face to face discussion show that discussants tend to form more developed arguments in online meetings [19].

Other studies have focused on particular design contexts. For instance, Friess investigated the extent to which novice interaction designers use evidence to support their design decisions [11]. She found that despite rigorous knowledge of user-centered design theory, in practice, only about half of the designers' claims were based on evidence, and most of the evidence was speculative. Mentis et al.'s study of group decision making in emergency planning showed that newly formed groups focus on refining particular ideas, whereas established groups first develop common ground and shared values [21]. Kriplean et al. describes the policies that structure collaboration in Wikipedia "talk pages" [17], finding that conflicts escalate by starting with requests for comment, and if necessary, requests for arbitration.

While studying online software design discussions is a new topic, argument and critical discourse has a long history in other disciplines. For example, the field of rhetoric seeks to inform how people persuade through speech and writing. The earliest known writing on the subject comes from Aristotle's *Rhetoric*, who claimed that there were three means of persuasion: establishing one's credibility, accounting for the emotional disposition of the audience, and using inductive and deductive logic. In this writing and those following it, scholars have named several types of rhetorical devices that help one achieve Aristotle's forms of persuasion. *Metaphor*, for example, compares two things by speaking of one in terms of the other ("this design is a ball of mud"), where *allusion* is an informal reference to a famous person or event ("this patch fails in Vista-like proportions"). There are several more obscure rhetorical devices, such as *sententia*, which is concluding with pithy wisdom ("I think the label is fine, but the user is not like me."), or *exemplum*, which is citing an illustrative story ("When I last used the factory pattern, no one knew how to instantiate. It was a disaster."). More recent work on argumentation [18] focuses on kinds of *claims*, such as *fact-based* evidence (which comes from a source independent from the person making the claim), *non-evidence* (circular reasoning), and *pseudo-evidence* (speculation and hypothetical reasoning).

In sum, there is substantial work on argumentation and group discussion in general, but little specifically about software design. Our study lays a foundation for this understanding.

3. METHOD

To understand the structure and content of open bug report design discussions, we gathered the complete set of closed reports from three online software projects, sample the most "contentious" of these reports, and then qualitatively analyze these reports for trends in process, ideas, rationale, and decisions.

We obtained our data set from the Bugzilla repositories of three online software projects: *Firefox* (a popular web browser), the *Linux kernel* (an operating system kernel), and *Facebook API* (an API for developing social networking applications). It is worth noting that these projects are quite different in their organization. Firefox is not only a large group of online contributors, but benefits greatly from the Mozilla corporation, which has several collocated developers in physical offices. Linux, like Firefox, is supported by corporations, in addition to many remote contributors. Facebook API is not an open source project, but does have an active community of developers who use the API and submit change requests to the open bug repository. Finally, in all of these communities, developers are not the only commenters: testers, designers, and users are known to contribute to these conversations.

Our goal in analyzing these reports was to understand *design discussions*, but bug reports cover a much broader spectrum of collaboration, including reproduction, code review, and repair [4]. In order to focus our analysis on *discussion*, we focused on reports of problems that had been reproduced and decided upon, by downloading only those marked as RESOLVED, VERIFIED, or CLOSED and resolved as FIXED, INVALID, or WONTFIX. We only considered reports available as well-formed XML from each project's website (a standard feature of each site's Bugzilla repository).

Our next step was to select a subset of the reports to read and analyze. We considered two approaches: sampling randomly and reading discussions varying in length, or sampling only reports with substantial discussion. We chose the latter, since our goal was to understand argument, disagreement, and persuasion, which we found was less common in more routine reports. Our first approach to identifying contentious reports was to use a word count, but this was biased by the predominance of error logs included in report comments. Instead, we used a measure of "contentiousness" based on the frequency of personal pronouns, counting occurrences of *I*, *you*, *we*, *they*, and *us*, and the phrases *IMO* and *IMHO* (given their prevalence in discussions). These words and phrases tend to indicate social involvement in a discussion [22]. We then ranked the reports by this measure, resulting in a power law distribution. We then randomly sampled 100 reports from the top 300 reports, which was the "elbow" of the power law curve. We chose 100 reports because of the time required to codify its ~1 million words.

Descriptive statistics about the reports read appear in Table 1. Firefox reports tended to be more contentious than reports in other projects, and are therefore a larger proportion of our data.

project	# of reports retrieve	# of reports read	# of comments read	report resolutions INVALID	WONTFIX	FIXED	[min – max], median of reports read # comments	# commenters	# duration (days)
Firefox	10,547	81	11,147	6	17	58	[7 – 526], 67	[2 – 83], 17	[1 – 2,272], 456
Linux kernel	898	12	629	4	0	8	[9 – 185], 43	unavailable	[1 – 1,464], 151
Facebook API	1,802	7	569	1	0	6	[20 – 212], 70	[3 – 25], 9	[2 – 406], 119
total	**13,247**	**100**	**12,345**	**11**	**17**	**72**	**[7 – 526], 63**	**[2 – 83], 16**	**[1 – 2,272], 356**

Table 1. For each repository, the number of reports downloaded and read, the number of comments read, the frequency of report resolution types, and descriptive statistics about the number of comments, number of commenters, and report duration.

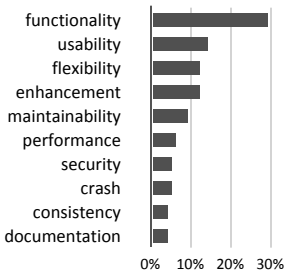

Figure 1. Qualities referred to in report titles.

code	definition	representative quote from sample
scope	defining what can be changed	neither of these bugs is about editing - editing is bug 255255.
idea	a description of a change	Really there needs to be 2 mechanisms for site icons. One for bookmarks and one for pages visited ...that are not in bookmarks.
dimension	factors affecting a change	So what are the differences between HTTP, HTTPS and FTP that a user should care about?
rationale	grounds for a particular opinion	I think that making part of the bookmarks listing dynamic while leaving the text static is less "clean" from a UI perspective.
process	comments about the discussion	We should have a clear rationale before laying hands on code. We should not be experimenting without a falsifiable hypothesis, if not a theory.
decision	event marking the closing of a report	[...] this bug has become the 'I lost my bookmarks' dumping ground, I'm going to resolve it as INVALID.

Table 2. Codes used to classify utterances in the comments, with definitions and representative quotes.

To see the distribution of report topics, we classified the report titles by the primary software quality referred to by the report author (though most report discussions contained references to a variety of software qualities). The results are shown in Figure 1. *Functionality* dominated report topics (generally referring to unintended behaviors), with *usability* improvements, requests for more *user flexibility*, and *enhancements* making a large proportion of other reports. The difference between "bugs" and "requests" was not always clear, as for many software qualities, this was a matter of opinion and prior intent by project leaders.

To analyze bug report comments, we used an inductive analysis approach [12], formulating our descriptions of the data in a systematic manner. We began by randomly assigning half of the reports to the 1st author and the remaining half to the 2nd, and first reading the reports in detail, trying to understand the discussion around design ideas, support for design ideas, and their merits. Most of the bugs we read contained significant discussion about bug reproduction, followed by discussion about potential fixes and their merits. Once we had achieved this more global understanding, we then focused on conceiving and iteratively refining a set of codes, which we then used to more rigorously classify the data. This process led to a list of potential concepts and structures, which we then distilled into the codes listed in Table 2. With these codes established, the two authors then read half of the sample independently, applying the six codes to each utterance in each report comment. We assessed agreement informally throughout.

To demonstrate how we classified statements, consider the abbreviated design discussion in Figure 2. By reading the left column, containing codes from Table 2, one can see the general progression from discussion of scope and ideas to rationale and decision. Such analyses form the basis of our results.

title	After shutting down Phoenix the favicon for the website www.helixcommunity.org is no longer present ...
scope	renaming this bug to **cover all favicon lossage**...
rationale	...it would be understandable to have this happen when the browser crashes, etc, but just shutting down Phoenix normally and then restarting it causes the favicons to be lost sometimes as well. **This is much more frustrating**.
idea	Wouldn't it be better to **store favicons seperatly** [sic] ... ?
rationale	**Why is this bug of 'minor severity'? I find it very annoying.** Since it's completely reproducible, applies to all new versions and has 30 votes...it's importance should go far up.
process	[*name*], the **severity rating is not a measurement of priority**. Severity reflects how much this affects operation or performance of the program. See http://...
process	I didn't know that. **How come priority isn't set** then?
process	**Priority isn't really used** within Firebird, targets are...
rationale	The main issue I have ... is that when I open the bookmarks menu, Firebird attempts to redownload the favicon and thus **asks me for username and password for this site, even when I don't intend to go to that site**.
scope rationale	**IMO, this is a site bug**. it should not be serving up favicons over SSL. what's the point? ... if we store it on disk, then **we are violating our policy of not storing "sensitive" data** on disk without explicit user consent.
scope idea	It's not a bug in the site [*name*] - favicons live at the root of the domain by default ... I wonder if it's worth creating a separate, **less volatile caching mechanism** for favicons...
scope rationale	... **who morphed this to be just HTTPS**? the "after shutdown" loss case **also happens in cases** of improper shutdowns/crashes where we flush the cache completely.
dimension	favicon fetching is an MS hack...**What does IE do?**
idea	OK, I seem to have gotten everyone excited! :) Perhaps all of these problems would be best solved using the technique of **saving a data: URL in bookmarks.html**.
dimension idea	BTW, i don't know that the data: solution is so ideal. what **happens when the site changes its favicon**? if we go with a solution that **walks the list of bookmarks** ...
idea rationale	It would be a useful to write **an extension which scans** your bookmarks looking for updated icons, but I wouldn't use it. **I've got a few hundred bookmarks - I'm sure I'm not alone**.
rationale	Well, the favicon has an URL, **whereas the title of the page does not**...I don't exactly see the parallels between the two.
rationale	...my point about it being difficult to update data: URLs is **apparently moot since [*name*] says he has a patch**! ;-)
rationale idea	...you are focused on the implementation, **not what the user sees**. The "least surprising behavior" here (IMHO) is when you drag the little favicon from the address bar to your bookmarks (or whatever), **the title and icon go with it and stick forever until you change it**...
decision	**Patch went in** for bug 174265 that should fix this.
process	It is fixed. **Do not change the status.** All remaining issues are being addressed in other bugs.

Figure 2. Selected discussion of a Mozilla bug report, tagged with codes from Table 2. Statements in bold are related to codes on the left; rows separate comments.

4. RESULTS

Overall, the design discussions in our sample varied widely in topic and exhibited a range of perspectives. In this section, we discuss the trends in the structure and content of these discussions by considering each of the six concepts from the coding scheme listed in Table 2. In each section, we cite representative quotes from our sample, using []'s to represent redacted names and elision.

4.1. Establishing Scope

One significant observation from our analyses was the importance of establishing a scope for discussion. Bug reports are primarily work items and not places for discussion, and so commenters often needed to specify types of changes to the software might be considered in the report.

We observed commenters establish two kinds of scope. One was under what *time horizon* the change would be completed (generally the current or next release). Commenters acknowledged the relevance or importance of related design ideas, but gave priority to certain fixes and deemed others as "off-limits," because such ideas were too difficult to implement given the time or technical constraints, required more deliberation, or were not backed up by enough evidence. For example:

> […] looks like a useful and meaningful revision to the Location Bar […] On the other hand, eTLD+1 and eTLD+2 highlighting are arguably problematic anti-phishing UI proposals and can easily be considered out of scope.

> Better presentation of URI elements would help http, but those were shouted down in another bug. We need to revisit that and gather hard evidence to better support that type of presentation, but that's way way out of scope for this release.

Another aspect of scope was the *generality* of the proposed solution, which ranged from changes of small scope such as hacks and workarounds to full redesigns of a feature. Commenters' preference was mostly for local, iterative changes, even if a more general fix would enable features or be more elegant. These iterative mindset was driven largely by technical and scheduling dependencies:

> […] there's no solution to this problem at the moment. Ultimately the prettyprint approach makes more sense to me, since it doesn't disturb the apparent DOM and more cleanly separates chrome from content. But we need the security model to change first, and although folks have been talking about doing just that, it's not going to happen before 1.5 […]

Scope also implicitly constrained what ideas, and what justification for ideas, were considered off-topic:

> I had though[t], probably wrongly, that this was part of the same 'bug' being discussed here.

Nevertheless, as we discuss later, commenters did comment out of scope and other commenters worked hard to keep comments within the established boundaries.

4.2. Proposing Ideas

Commenters design ideas are what usually sparked significant discussion. For example:

> I would suggest that the order should not be determined by what was VISITED, but by what was TYPED into the location bar or SELECTED from the drop down of the location bar.

This idea was a revision to the Firefox ranking of auto-completed URLs. From the perspective of design ideas, discussions tended to gravitate around small deviations from the current design along different qualities. For example, from the same report as above:

> I suggest an option in the preferences […] which would decide, which sorting algorithm should be used. There are several possibilities, all of them wanted by someone else.

This idea focuses more on *user flexibility*, whereas another focused on *consistency* with other browsers:

> That list ought to be sorted by last visited, just like in IE.

Like the examples above, most discussions were trajectories through the design space of a feature, with debate about the impact of various ideas on different software qualities. Some ideas were workarounds, in which the system did not change at all, but the inputs to the system changed considerably. In other cases, there was justification to leave the system unchanged. There were frequently proposals for major redesigns of features, but that were rarely considered unless part of the larger plans for a release.

Ideas were generally described with words, except when they involved user interfaces. These ideas generally moved from verbal descriptions, as in the above quotes, to representations with more structure, such as ASCII mockups as in Figure 3, to more formal photoshop sketches, and eventually a code patch. All ideas were iterated, especially code, which was frequently reviewed.

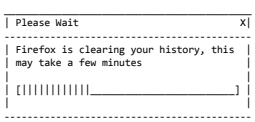

Figure 3. An ASCII art design idea, from a Mozilla report.

4.3. Identifying Design Dimensions

Throughout discussions, commenters raised a number of questions to help understand the relationship between different parts of the software, as well as the behavior and expectations of people. In our analyses, we viewed these questions as dimensions of a design space, since the subjects of commenters' questions were aspects of the software or users that had the potential to vary and interact. For example, some posed questions about the behavior of *users*:

> Does the user expect the favicon to remain stored, or to be updated when the site icon changes?

> Are security-savvy people going to get phished?

> can we get away with having UI that's only visible to people who can clearly distinguish red from black?

These questions refer to dimensions such as user expectations, the degree to which security experts notice security problems, and the vision capabilities of a user base. Similarly, commenters identified properties of the *software*:

> I don't know offhand if this would affect the slow startup.

> Does anyone have any idea what the effect of the doubled entries in the registry will be?

> will fixing this bug or fixing bug 47903 cause more problems […]?

These questions referred to dimensions such as how much performance depends on a particular feature, how an operating system deals with a particular form of data, and the likelihood of dependencies between one feature and other.

Commenters sought to identify these dimensions usually in order to establish constraints on design choices, or to further inform the group's understanding of certain software qualities, such as the *usability* or *performance* of a particular design. Commenters did not always get answers to these questions, but these questions did compel commenters to reply with additional questions.

4.4. Defending Claims with Rationale

The most dominant type of comment in design discussions was commenters' *rationale* for their design ideas. In our analysis, we identified two fundamental aspects of rationale: the (1) *software quality* to which it appealed and (2) a *rhetorical device* to reinforce the quality's importance. Overall, when arguing for a particular idea, commenters tended to support a single quality at a time (rather

quality	#	representative quote from sample
consistency	160	For years and years **backspace has been _the_ way of going back in history**. Please, please, please, changing the behavior of backspace has, in my opinion, crippled the browser [...] Data loss is a non-argument.
flexibility	131	IMO not a great usability experience. I believe **a user should have the option** to open the new tabs right where he is at that point
annoyance	87	By stating that this file is a particular type, the user's reasonable question is, "Then why don't you handle it like one?" Until Mozilla can do a better job of answering that question [...] **user frustration will continue**.
user efficiency	75	When opening a new tab, **nobody wants to be forced to scroll potentially dozens of tabs away** from the current position/tab all the way to the very far right end of the line of open tabs. As a default behavior, that's just absurd and not just a little frustrating.
security	70	We should only impose this security theory on the Web at large **if we're sure it constitutes a clear security improvement**, which we definitely have no consensus on. If we must argue from authority, the final word will not come from the UI designers.
clarity	64	I think using &pref.singular; on those 2 cases in prefs is not a good idea, as these are 2 of 3 selectable drop-down menu options (thus choices, not settings) of which the 3rd is still left as it is. Doing so would **look weird or even be confusing.**
visibility	61	The thing I liked about the Options dialog was that **it showed clearly a number of choices that were available**, not hidden by a dropdown. I sort of cringed to have to show this to everyone the first time.
feasibility	61	After a little more thought, **scratch "hard" and replace with "not *too* hard"** [...] we can cobble something together.

Table 3. Software qualities appearing > 45 times and frequencies and quotes for each. Bold indicates software qualities identified.

than discuss tradeoffs between multiple qualities) and commenters provided little evidence for their quality claims.

To illustrate these two aspects, consider this example, which uses hyperbole to argue for visibility:

> **Only by** [hyperbole] enlarging the padlock notion into a more complete security display can this **be made obvious** [visibility] ...

Because there was considerable variety in the types of qualities and rhetoric that commenters stated, we subdivided the rationale code in Table 2 into specific types of software qualities and rhetorical devices employed by commenters. To do this, both authors independently scanned statements tagged with the *rationale* code and generated labels of the types of qualities and devices used, then merged their lists and settled upon a small set of codes. Each author then applied the codes to each *rationale* statement in the sample.

With regard to software qualities, there was a bimodal distribution of frequencies, the most common of which appear in Table 3. These include *consistency*, *annoyance*, and *flexibility*, among others (whereas the infrequent qualities included, in decreasing frequency, *functionality*, *simplicity*, *maintainability*, *guessability*, *performance*, *utility*, *aesthetics*, *reliability* and *bloat*). Obviously, this ranking is highly influenced by our sampling approach: these qualities could be intrinsically contentious or the projects we studied may place a high importance on these qualities. Interestingly, the list is largely devoid of *internal* software qualities such as *maintainability* and *code aesthetics*, suggesting that commenters were more concerned with implications for user experience than for technical constraints, at least in contentious discussions.

A more revealing aspect of commenters' rationale were the *rhetorical devices* that they used to persuade each other. As with the software qualities, the frequencies of different devices exhibited a bimodal distribution, the most common of which appear in Table 4. The most popular rhetorical devices were *anecdote*, *speculation*, *generalization*, and *hyperbole* (whereas the infrequent devices, in decreasing frequency, included *hypothetical*, *insult*, *priority*, *statistics*, *policy*, and *sententia*). With regard to anecdotes, it was common for commenters to tell stories about friends or family in order to illustrate some point of view, then generalize this story to a larger population. It was also common for commenters to use the word "user" in an elastic way [7], describing users in whatever way would support the commenters' argument:

> Joe user won't have the knowledge, nor the need to acquire it, and is unlikely to shoot himself in the foot using a feature he doesn't know exists.
>
> Users who don't know about URLs don't write HTML [...]

Most other forms of argument were *non-evidence* and *pseudo-evidence* [18], devoid of objective justification (though not necessarily wrong). The comments above for example, are *generalizations* with a certain face validity, but no objective verification. *Connotation* and *hyperbole* were also popular forms of non-evidence:

device	#	representative quote from sample
anecdote	176	For example, **a teacher at the high school I went to** [...] could be seen to do things that, well, are looked upon badly by technical people. For one, she could be seen numerous times just clicking "OK" on a dialog, and shortly after asking why thing weren't working.
speculation	114	**I am pretty sure that** the vast majority of users will not know or care what is in the sublist.
generalization	112	For 99.9 percent of Firefox's installed base, **they DO NOT CARE about having multiple versions installed side by side**. I know that IE doesn't give users this option, so to the lay person [...] they will not miss the ability [...] because they had no idea you could do that.
hyperbole	87	You mentioned this before and I still think it is rather unacceptable to leave such a critical bug in released versions. **It renders Firefox 1.x rather almost unusable** - I know I am exaggerating a bit, but I guess you get the point.
pragmatism	73	I'm not very fond of it either, but it's probably the only way we can do what this bug asks for. Frankly, I'm much **rather not take this (rather large) risk**, and instead just change existing users and new users over to something similar to the current Firefox start page.
impact	71	But this spaminess thing is actually hurting people by being broken or imperfect. **It is impacting my business when clients want their money back** because their app is blocked, or when I can't show off a cool app I built in my portfolio because it has been killed.
logic	46	That is a **false analogy**. A bookmark manager may have similarities to a file manager, but in this case your analogy is an informal fallacy for this argument. It's also an **ignoratio elenchi** because it has nothing to do with implementing find bookmark like Netscape
connotation	46	Seems very **intuitive** to me.
tradeoff	45	Hippocrates' instruction is good, but I think too absolute for us. We've always known, and I recall receiving sage guidance from you to this effect :), that we sometimes have to **take some harm to a given use case to improve one that's of greater value**.
authority	37	**In person [*name*] said to me "if we don't believe in the design we shouldn't ship it."**

Table 4. Rhetorical devices appearing > 30 times and frequencies and quotes for each. Bold indicates use of rhetorical device.

Definitely an **annoyance**.

Here, the commenter simply referred to the concept of annoyance and let exaggeration play a persuasive role. Most uses of non-evidence were characterized by a narrow view of design, appearing to value only a single quality, as opposed to a more nuanced view of design tradeoffs. The above devices were used primarily to defend and critique ideas, whereas commenters relied on *pragmatism*, *logic*, *impact,* and *authority* to build consensus and resolve disagreement:

> [I would be convinced by] An argument that demonstrates that this is a feature that will be beneficial to a majority of users and just a small percentage. [...] I won't close this without discussing it with one of our user experience folks as well.
>
> The fact that this bug has been open for six years and only has 32 votes indicates that it's not a big issue for most people.
>
> [...] what I propose is a pragmatic and simple way of not breaking backwards compatibility [...]

A minority of commenters referred to tradeoffs and priorities, raising the level of discussion from isolated design ideas to the space of design ideas:

> They all matter... If we [make the change], we will make security exploits easier, not harder, and we will make diagnosing other real-world problems harder, not easier.
>
> Sometimes hunches, intuition, and aesthetics are the right way to judge a change, but it depends on what the downsides are. I think there's a fundamental disagreement on the level of risk changes to the presentation URIs incur.

Such comments were often viewed as didactic and unhelpful by commenters supporting a particular design idea.

4.5. Moderating Process

While many discussions proceeded unmoderated, discussions across all three projects often crossed community-defined boundaries of acceptable content. In most cases, this occurred when the alleged *impact* of the bug led to intense disagreement. Commenters with authority or those that were willing to take lead tried to control the process to maintain focus on how the report should be resolved.

Some commenters expressed frustration, often marked by *hyperbole*, over the lack of attention to a particular issue or the whole bug itself:

> There is something flawed in your methodology, guys. There just is. There are at least 20 or more bug reports on this issue, most with a dozen or more people chiming in. We can't all be wrong.

Developers tried to appease the situation by providing timelines or explaining the difficulty of devising a solution:

> I understand the frustration and want to reassure all of you that it is high on our bug priority list. The solutions are not exactly trivial so bear with us while we figure it out.
>
> OK, calm down everyone. How many times do I have to say it? It's what we want to try out to start with. That is not code for "we've made a decision" or "your arguments all suck and we're going to ignore them".

Other process-related comments were about moving unrelated discussions elsewhere, because the scope of the conversation had moved beyond the scope of the report. These included comments on enhancement requests or new features that were not directly relevant and added to the complexity of the report resolution:

> While some interesting philosophical ideas have been brought up..., I'd like to point out that improvements, overhauls, and philosophical ideals should be moved to either a discussion on the forum, or into an enhancements entry here... This thread is no longer relevant to the original situation, and should not be used as such.

Commenters also tried to enforce etiquette to keep the discussion on track. For example, in some cases the medium of the conversation (a sequential list of comments) led to problems with missing context:

> [...] but from here on out, can we refrain from spamming the bug with more comments along the lines of "it happens to me". That is the purpose of the voting mechanism in bugzilla. PLEASE read a bug fully before posting a new comment to it, and only then if there is information to add, and you aren't repeating something that is already there.

In this example, this request was in the middle of hundreds of comments and most commenters did not notice it.

4.6. Making Decisions

Given the results in the previous section, a key question is whether any of rationale expressed by commenters influenced the actual *decisions* made for each report. Did commenters' arguments influence whether a report was resolved? Did they influence *how* a bug was fixed? Whose ideas were implemented and to what extent did they incorporate the other commenters' perspectives? In our analysis, it became evident that there were three distinct *decision patterns*, with differing levels of commenter influence. We classified each report as one of the three.

The most common pattern (63% of reports) involved just developers (where developers were considered anyone whose e-mail addresses appeared in the assigned-to field of a bug report). These began with a brief discussion around the functional design for the change, which quickly led to consensus and the implementation of a patch. The discussion in these reports then focused on the design of the *code* for the patch. A key characteristic of these reports was how little of the functional design space was explored: these reports usually identified few alternative designs and involved only minor tweaks to maintainability, aesthetics and performance. Though developers would raise many other qualities in these reviews, pragmatism and local, iterative changes dominated decision making. Issues of impact and generality were deferred by creating new bug reports to represent their concerns.

A different pattern (19% of reports), involved largely *divergent* discussion and usually involved both developers and users. There were two kinds of outcomes of these discussions. In a third of these, a developer would end the discussion with a decree, sometimes offering their rationale. When they did offer rationale, it was largely based on *pragmatism* and *impact* (as in Table 4):

> Eventually someone will write an extension to allow this, but the percentage of the user population that knows what application/octet-stream means is small. Very small.

It was more common in the Facebook reports that developers would explain the rationale for a change. Firefox developers were more likely to state their decisions without rationale, while other commenters tried to overturn the decision:

> Overall an outrageous development - if the developer(s) are so insistent that this is not the desired behavior (which I disagree - there is wide agreement that there should be a one-button shortcut to go 'back' and, like it or not, Backspace has emerged to fill that role), then the bug should be marked WONTFIX.

The other two thirds of divergent discussions ended because some independent change to the software made the discussion moot. For example, there were many contentious discussions about features of the Firefox location bar, most of which were made obsolete by a new implementation that arrived over a year later:

> [...] Location Bar automcomplete has been redesigned from scratch in Firefox 3. It uses a much more elaborate sorting algorithm and order visited is only one of many critieria.

The third decision pattern we found (18% of reports) involved *convergent* discussion, usually between developers and users, which typically led to developers vetoing a proposed change. These

vetoes rarely came with rationale, but when they did, they usually had to do with inconsistency with prior decisions, pragmatism, and authority. In particular, there was frequent debate about distinguishing between *officially supported* uses of the software and *unsupported appropriations* of the software. To the idealists in these discussions, there was an original intent to the design of the software, and whether or not users found other uses for features, developers ought to be free to change the system in ways consistent with its original design. The opposite and more pragmatic viewpoint was that regardless of the original intent, users are what drive adoption of the software, and unexpected appropriations should *become* supported.

On either side of these debates, uncovering the original intent of a design was a crucial part of the discussion. For example, idealistic commenters would find design documents or IRC conversations between project leaders and use the rationale in these sources to document officially supported uses. One debate took place in a Facebook API report, in which users of the API were concerned about a change to the spam detection algorithm, and its reclassification of their Facebook app as spam, thus lowering their ad revenue. Debate centered around the original intent of spam detection algorithm and whether the redesign was consistent with it.

Overall, it was clear that the most powerful factors in decision making were *authority* (of developers over users) and *action* (writing a patch). Furthermore, developers usually used their power to make pragmatic decisions that addressed the system in actual use by users, rather than the *ideal* system sought by some commenters. The only cases in which design decisions were influenced by design *discussion* were in reports with a small number of developers, in which they briefly discussed the functional design for a change. One commenter summarized it nicely:

> Seems like this issue is just a matter of different personal opinions. Personal opinions of those who have power to fix this prevail, by definition.

5. DISCUSSION

Our results indicate that the commenters in our sample were passionate about making decisions that would positively impact users overall, but that they often lacked the information necessary to do this in an objective manner. The high reliance on anecdotes and speculation and the rare discussion of tradeoffs and priorities is consistent with prior work on argumentation [18,5], including user-centered design [11]. However, our results also suggest that the lack of evidence in these discussions was due more to the difficulty of gathering data about users, than an inherent lack of rationality. In this section, we discuss these results, their implications on our understanding of software design, and their impact on the design of computer-mediated design discussion tools.

5.1. Discussions as Design Space Exploration

Our primary question in this study was *how are design decisions discussed in open bug reports*? Our results suggest that these discussions are essentially explorations of design spaces, heavily constrained by the scope of the report, the sequential medium of the bug report comments, and the authority of its participants.

To illustrate this idea, consider Figure 4, which portrays the design space for the discussion in Figure 2. The space is composed of several dimensions (two of which are shown). In this figure, the circle in the center is the current design of the system, and the smaller points extending from this large circle are the proposed changes to the system design, moving along different quality dimensions. The discussions in our data essentially served to map design spaces such as the one in Figure 4, but with words rather than drawings. Moreover, rather than the discussions serving a deliberative purpose as they do in other domains [26], explicitly enumerating design alternatives, design tradeoffs, and design priorities, the discussions we observed were ad hoc and implicit in their consideration of these aspects of the decision. A minority of contributors alluded to tradeoffs and priorities, but the discussions rarely considered them explicitly. One possible explanation for this is a lack of design experience on the part of many of the commenters; as many of the commenters were testers and users, rather than developers, this would not be surprising. An alternative explanation supported by our data is that developers attempted to suppress design debate, as the bug reports were not the place to reconsider design decisions.

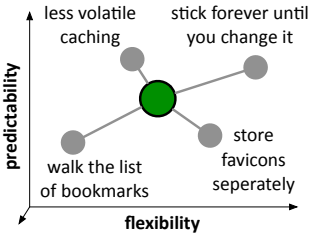

Figure 4. Design ideas discussed in Figure 2 along two dimensions.

5.2. Pragmatism, Idealism, and Dependencies

One issue orthogonal to the level of deliberation in discussions were commenters approaches to resolving design tradeoffs. In particular, we observed that most debates revolved around the conflicts between existing and future uses of the software and deciding which would be supported. For instance, in one report, commenters discussed whether to revise a preferences dialog, but recognized that since users had learned the location and function of a particular preference in the Firefox dialogs, future changes would be risky, because any change could confuse existing users. Most of the discussions dealt directly with these design conflicts, drawing upon anecdote, pragmatism, and impact arguments to help separate the intended uses of software unintended ones.

These decisions about intended use appear in a variety of other design contexts. For example, when the dominant culture of a software team is pragmatism, we would predict that early design decisions are "sticky," and later changes to a system tend to be iterative rather than transformative, because of the risk and cost of breaking user dependencies [6]. Alternatively, when decisions are primarily based on ideals and policy, user dependencies are frequently broken in order to preserve the underlying values of the ideas. Take, for example, Apple, which is known to completely re-implement APIs to innovate and ensure consistency, but at the expense of dependencies that people have on legacy code.

5.3. Software Quality and Measurability

Our results show that underlying the ambiguity around design tradeoffs and intended use were struggles to characterize a variety of software qualities. In particular, our data show that commenters referred frequently particular software qualities, but rarely to evidence supporting their claims about these qualities. The result of these differences are apparent when comparing *measurable* qualities from *immeasurable* ones. For example, in Table 3, the most frequently discussed qualities are qualities that are generally difficult to assess and compare. In contrast, qualities such as performance were rarely debated: either a revised algorithm was faster or it was not. Moreover, we observed little explicit discussion of which qualities had priority in the community; for example, in the Mozilla discussions, we expected a particular focus on flexibility, security, and openness, but did not observe these qualities taking priority over other qualities. These findings suggest that a critical part of supporting online software design discussions is not encouraging explicit discussion of competing software qualities, but articulating which take priority.

5.4. Limitations

Our study has several limitations. For example, because we read only contentious reports, and we selected these reports based on a certain notion of contentiousness, we may have overlooked other types of reports with other forms of argumentation and process. Furthermore, because we only read design discussions in bug reports, and not mailing lists, internet relay chats (IRC), discussion boards, and other venues, we cannot be sure that the form of argument we observed in bug reports generalizes.

Our focus on Firefox, Linux, and the Facebook API may also have biased our data. While these three represent a diverse sample, organizational cultures with different conventions, etiquette and forms of authority may influence design discussion. For example, in more conventional corporate hierarchies, we might expect fewer commenters but stronger social ties, leading to different forms of persuasion and decision making.

There are also issues outside our sampling approach. For example, we did not know the identities of the commenters in reports. We are confident that not all of the commenters were developers (many commenters stated so), and so it may be the case that certain types of argumentation are more common in non-developer populations. Also, we had to infer whether a commenter was a developer just based on context, which may have been inaccurate. Furthermore, these findings are not necessarily representative of decision-makers in these developer communities. For example, perhaps the developers making decisions about patches *ignore* contentious reports and only accept those for which there is broad consensus.

6. CONCLUSION

Our results illustrate that in addition to the usual challenges of understanding a design space and making tradeoffs between competing qualities, contributors to open bug reports struggle with a number of issues. The inability to *measure* many important software qualities led to extensive use of anecdote, generalization, and speculation. The fundamental choice between achieving original design intents and adapting to user needs made bug reports a hotbed of conflict between developers and users. Finally, temporal presentation of discussion comments were inadequate for the proposals and critiques that dominated discussion.

These findings have several implications. First, online discussion tools should be redesigned to make proposals and critiques in design discussions more explicit. Second, our results raise several questions about the role of authority, prior intent, and the measurability of software qualities in design discussion. These and other issues are central to improving the quality of open software.

7. REFERENCES

[1] Aranda, J. and Venolia, G. (2009). The secret life of bugs: Going past the errors and omissions in software repositories. *Int'l Conf. on Software Engineering* (ICSE), 298-308.

[2] Bach, P.M. (2009). Design information sharing across multiple knowledge systems in a FLOSS community. *iConference*.

[3] Barcellini, F., Detienne, F., Burkhardt, J.M., Sack, W. (2008). A socio-cognitive analysis of online design discussions in an open source software community. *Interacting with Computers*, 20, 141-165.

[4] Bettenburg, N., Just, S., Schröter, A., Weiss, C., Premraj, R. and Zimmermann, T. (2008). What makes a good bug report? *Foundations of Software Engineering* (FSE), 308-318.

[5] Canary D.J., Brossmann B.G., and Seibold D.R. (1987). Argument structures in decision-making groups. *Southern Speech Communication Journal*, 53(1), 18-37.

[6] Christensen, C.M. (2003). *The innovator's dilemma*. Harper Collins.

[7] Cooper, A., Reimann, R. and Cronin, D. (2007). *About face 3: The essentials of interaction design*. Wiley Pub.

[8] Cummings, J. and Kiesler, S. (2005). Collaborative research across disciplinary and organizational boundaries. *Social Studies of Science*, 35(5), 703-722.

[9] Espinosa, A., Kraut, R., Slaughter, S., Lerch, J., Herbsleb, J. and Mockus, A. (2002). Shared mental models, familiarity, and coordination: A multi-method study of distributed software teams. *Int'l Conf. on Information Systems*, 425-433.

[10] Farnham, S., Chesley, H. R., McGhee, D. E., Kawal, R., and Landau, J. (2000). Structured online interactions: improving the decision-making of small discussion groups. *Computer-Supported Cooperative Work* (CSCW), 299-308.

[11] Friess, E. (2008). Defending design decisions with usability evidence: A case study. *ACM Conf. on Human Factors in Computing Systems* (CHI), 2009-2016.

[12] Glaser, B. G. (1992). *Emergence vs forcing : Basics of grounded theory analysis*. Mill Valley, CA, Sociology Press.

[13] Gutwin, C., Penner, R. and Schneider, K. (2004). Group awareness in distributed software development. *Computer Supported Cooperative Work* (CSCW), 72-81.

[14] Hiltz, S., Johnson, K., and Turoff, M. (1986). Experiments in group decision making: communication process and outcome in face-to-face versus computerized conferences. *Human Communication Research*, 13, 225- 252, 1986.

[15] Ko, A. J., DeLine, R. and Venolia, G. (2007). Information needs in collocated software development teams. *Int'l Conf. on Software Engineering* (ICSE), 344-353.

[16] Ko, A.J. and Chilana, P. (2010). How power users help and hinder open bug reporting. ACM Conference on Human Factors in Computing Systems (CHI), Atlanta, GA, USA, to appear.

[17] Kriplean, T., Beschastnikh, I., McDonald, D. W. and Golder, S. A. (2007). Community, consensus, coercion, control: CS*W or how policy mediates mass participation. *ACM Conference on Supporting Group Work* (GROUP), 167-176.

[18] Kuhn, D. (1991). *The skills of argument*. Cambridge U. Press.

[19] Lemus, D.R., Seibold, D.R., Flanagin A.J., Metzger M.J. (2006). Argument and decision making in computer-mediated groups. *Journal of Communication*, 54(2), 302-320.

[20] Li, Q., Heckman, R., Allen, E., Crowston, K., Eseryel, U., Howison, J., and Wiggins, A. (2008). Asynchronous decision-making in distributed teams. *Computer Supported Cooperative Work* (CSCW), San Diego, CA.

[21] Mentis, H.M., Bach, P. M., Hoffman, B., Rosson, M.B., and Carroll, J.M. (2009). Development of decision rationale in complex group decision making. *ACM Conf. on Human Factors in Computing* (CHI).

[22] Pennebaker J.W., Mehl M.R. and Niederhoffer K.G. (2003). Psychological aspects of natural language use: Our words, our selves. *Annual Review of Psychology*, 54(1), 547-577.

[23] Seaman, C. B. and Basili, V. R. (1998). Communication and organization: An empirical study of discussion in inspection meetings. *IEEE Trans. on Soft. Engineering*, 244(7), 559-572.

[24] Straus, S. G., McGrath, J. (1994) Does the medium matter? The interaction of task type and technology on group performance and member reactions. *J. of Applied Psychology*, 79, 87-97.

[25] Twidale, M.B. and Nichols D.M. (2005). Exploring usability discussions in open source development. *HICSS*.

[26] Ullman, D.G. (2009). *The mechanical design process*, McGraw Hill.

[27] Zannier, C., Chiasson, M., & Maurer, F. (2007). A model of design decision making based on empirical results of interviews with software designers. *Information and Software Technology*, 49(6), 637-653.