A Qualitative Study of Mozilla's Process Management Practices

Olga Baysal and Reid Holmes
David R. Cheriton School of Computer Science
University of Waterloo, Canada
{obaysal, rtholmes}@cs.uwaterloo.ca

May 16, 2012

1 Informal Overview

The Mozilla Anthropology[1] project was started in late 2011 to examine how various stakeholders make use of Bugzilla in practice and to gain a sense of how Bugzilla could be improved in the future to better support the Mozilla community.

During this process, Martin Best interviewed 20 community members; we have split these 20 interviews into over 1,200 individual statements and performed an open card sort to try to gain insight into high-level themes about Bugzilla to identify strengths, weaknesses, and ideas for future enhancement of the platform. During this process, 4 high-level categories emerged from the data (along with 91 sub-themes). These were:

- Situational awareness (19 participants, 208 quotes, 14 sub-themes)
- Supporting tasks (20 participants, 700 quotes, 53 sub-themes)
- Expressiveness (20 participants, 188 quotes, 12 sub-themes)
- The rest (20 participants, 166 quotes, 12 sub-themes)

While some of the sub-themes were not surprising (e.g., Supporting Tasks::Reporting:: Submission process), the topics that emerged around situational awareness really stood out. While Mozilla dashboards are often thought of as quantitative tools (e.g., bug open/close charts, performance changes, etc.), a real desire for personalized list-based dashboards became apparent. For example, several users wanted the ability to privately mark bugs to watch (Situational Awareness::Private Dashboard::Watch List (10 participants, 18 quotes)), while others wanted to be able to gain insight into a developer's role within the

[1] `https://wiki.mozilla.org/Bugzilla_Anthropology`

project (are they active, are they module owners, do they have review rights, etc.) (Situational Awareness::Public Dashboards::Developer Profiles (12 participants, 31 quotes)).

Situational awareness also crosscut some of the sub-themes from other themes. For example, Supporting Tasks::Code Review::Recommending Reviewers (6 participants, 10 quotes), where developers really wanted the ability to determine the work load of a reviewer before requesting a review (this could be captured in a public dashboard that showed the reviewer's current queue).

Rather than chart-based views, most of the Situational Awareness shortcomings in Bugzilla could be implemented as lists that pivot or enumerate various pieces of metadata already presented in the system. By treating people as first-class entities within Bugzilla (bugs are currently first-class, people are just metadata on bugs), developers could modify a personal dashboard page that could have public and private information and serve as a heads up display showing them their own specific view of the Bugzilla repository.

We are continuing to analyze the categories that have emerged from the card sort to identify other themes that could suggest improvements to Bugzilla to better support the Mozilla community. Any insight into the categories and themes we have identified would be greatly appreciated!

2 Open Coding Approach

We have performed a qualitative study on the data collected by Martin Best[2] who conducted interviews with 20 Mozilla developers on their engagement and experience with Bugzilla bug tracking system.

We applied an open coding technique to classify 1,213 individual comments into naturally emerging themes. We identified 15 themes and 91 sub-themes, containing between two to 41 comments. Later, the 15 themes were grouped into 4 concept categories. These concept categories consist of two to six themes. Each concept category relates to a different aspect of how Mozilla developers interact with Bugzilla to report and resolve bugs and to manage the collaborative process.

We provide an overview of the concept categories, as well as their themes and sub-themes. For each theme and sub-theme we provided the number of individual participants commenting on a certain issue and the total number of quotes given. For each sub-theme we developed a synthetic quote that provides a general thought on the topic. Synthetic quotes are generated by combining participants' comments into a single statement. Due to space constraints, synthetic quotes are not reported in this document.

[2] `http://blog.mozilla.org/mbest/`

3 Concept Categories

The four concept categories that have emerged during the card sort do not necessarily have direct correspondence to the tasks developers perform daily. Rather, they are a combination of actions and mental activities developers perform when considering and executing these tasks.

Situational Awareness is a form of gathering information on what's happening on a project. Participants often find themselves trying to identify the status of a bug - what is is waiting on and who, who is working on what bugs, what are the workloads of others, who is the best person to review the patch, as well as trying to track their own tasks - how many bugs do I need to triage/fix/review/follow up.

Supporting Tasks addresses issues related to specific tasks such as code review, triage, reporting, testing, release management, etc.

Expressiveness provides means to communicate the process with others. Whiteboard and keyword tags are primary ways for tracking status, seeking approval, increasing awareness and interest, etc.

The Rest includes topics related to interacting with version control systems, other general issues about Bugzilla such as performance, culture, process, etc.

Table 1 describes themes and sub-themes that each of the 4 categories is comprised of.

Table 1: The overview of the concept categories.

Category/theme/sub-theme	**# Participants**	**# Quotes**
Situational Awareness	**19**	**208**
• Dashboards	18	99
– Public dashboards	16	60
Developer profiles	12	31
Workload transparency	12	22
Communicating interest	8	12
– Private dashboards	15	39
Tracking activity	12	21
Watch list	10	18
• Collaborative Filtering	4	8
– Highlighting important comments	4	8
• Status	18	56
– What is the current status?	8	24
– Next action	11	20
– Bug hand-off	6	12
• Email	17	45
– Overwhelming volume	8	12
– Email is important	11	16

Table 1 – *Continued*

Category/theme/sub-theme	**# Participants**	**# Quotes**
– Email is not important	2	2
– Filtering	6	9
– Formatting	4	6
Supporting Tasks	**20**	**700**
• Code Review	20	130
– Recommending reviewers	6	10
– Patches	10	27
– Importance of review timeliness	8	12
– States	12	32
– Process and community	8	11
– RiskReward	5	8
– Misc	17	30
• Triage & Sorting	20	259
– Sorting / filtering	15	35
– Bug assignment	12	25
Who gets the bug?	7	10
Self Assignment	6	10
Unassigned	4	5
– Components/products	17	41
– Component owners	3	4
– Last touched	4	7
– Is this bug actionable?	10	16
– Volume of bugs to triage	4	5
– Duplicates	4	7
– Bugs in General	6	9
– Bug kill days	3	4
– Triage & community engagement	5	7
– Midair collision	2	2
– Triage meetings	5	5
– Triage of other components	2	4
– Triage process	10	18
– Comments	4	4
– Misc	15	39
• Reporting	20	145
– Submission is harder / more confusing than needed	17	33
– Improving reporting	10	14
– Defects in the reporting experience	4	9
– Metadata	10	23
– The role of the description and summary fields	9	14

Table 1 – *Continued*

Category/theme/sub-theme	**# Participants**	**# Quotes**
– Possible enhancements to improve reporting	12	15
– External tools	2	3
– Intimidating for new users	7	8
– Misc	8	26
• Search	14	53
– Quick search	4	7
– Advanced search	5	5
– Saved/shared	5	8
– Hard/confusing	7	10
– Date range	3	3
– Performance	3	5
– Product	2	2
– Dups	3	5
– Misc	6	8
• Testing & Regression	14	37
– Regression	5	9
– Testing and its reliability	8	15
– QA and Validity	6	9
– Fuzzing	2	4
• Tasks	13	76
– Role-specific views	4	7
– Release management	10	27
– Where a bug lands?	10	22
– Statistics	8	12
– Workflow	6	8
Expressiveness	**19**	**188**
• Metadata	20	132
– Tracking flags	14	38
– Whiteboard	17	36
– Keywords	14	22
– Meta bugs	3	6
– Metadata (general)	8	12
– Tagging (general)	3	4
– Status flags	9	14
• Severity/Prioritization	19	56
– Unclear definition of priority/severity	6	8
– Priority	13	14
– Severity	11	16
– Prioritization	8	13

Table 1 – *Continued*

Category/theme/sub-theme	# Participants	# Quotes
– Misc	5	5
The Rest	**20**	**117**
• Version Control	8	43
– Reviewing via Github	3	12
– Checkins	6	13
– Branching	4	6
– Merging	2	4
– Integration of Bugzilla with Hg	4	8
• Bugzilla Issues	16	64
– UI	8	11
– Advanced scripts and tweaks	7	11
– Process	3	3
– Feature pages	8	11
– Performance	2	3
– Culture	4	6
– Misc	12	19
• Useless	9	10

4 About the Authors

Olga Baysal is a PhD student at the School of Computer Science, University of Waterloo, working under the supervision of Dr. Michael Godfrey. Her research interests include mining software repositories, software evolution, applying AI and IR techniques into the field of software engineering. She obtained her MMath degree at the University of Waterloo supervised by Dr. Andrew Malton.

Reid Holmes is an Assistant Professor in the Cheriton School of Computer Science at the University of Waterloo. His interests include understanding the cognitive aspects of software engineering, software reuse, example recommendation systems, and longitudinal dynamic analyses. He has published articles in top-tier publications including the International Conference on Software Engineering (ICSE), Foundations of Software Engineering (FSE), Transactions on Software Engineering (TSE), and Transactions on Software Engineering and Methodology (TOSEM). He has won distinguished paper awards at ICSE and FSE. He did a postdoc at the University of Washington advised by David Notkin, received his Ph.D. at the University of Calgary advised by Robert J. Walker, and received his M.Sc. at the University of British Columbia advised by Gail C. Murphy.