A Qualitative Study of Mozilla's Process Management Practices
Technical Report CS-2012-10

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

June 14, 2012

1 Informal Overview

The Mozilla Anthropology[1] project was started in late 2011 to examine how various Mozilla community 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 community.

During this process, Martin Best interviewed 20 community members; we have split these 20 interviews into over 1,200 individual quotes and performed an open card sort to gain insight into high-level themes about Bugzilla to identify strengths, weaknesses, and ideas for future enhancement of the platform. During this process, four high-level categories emerged from the data (along with 15 themes and 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 deltas, etc.), a real desire for personalized list-based dashboards became apparent. For example, several users wanted the ability to privately mark

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

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 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 categories. 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 Bugzilla 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 (SQ) are generated by combining participants' comments into a single statement. For a complete analysis, please refer to Appendix A.

[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 organizing information on the current status of 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 provides an overview of the categories, themes, and sub-themes.

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
– Email is not important	2	2
– Filtering	6	9

Table 1 – *Continued*

Category/theme/sub-theme	**# Participants**	**# Quotes**
– 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
– Risk/Reward	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
– Possible enhancements to improve reporting	12	15
– External tools	2	3

Table 1 – *Continued*

Category/theme/sub-theme	**# Participants**	**# Quotes**
– 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
– Misc	5	5

Table 1 – *Continued*

Category/theme/sub-theme	# Participants	# Quotes
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

Table 1: The overview of the concept categories.

4 About the Authors

Olga Baysal is a PhD student in the David R. Cheriton 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 static and 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.

A Appendix

A.1 Situational Awareness [19p, 208s]

A.1.1 Dashboards [18p, 99s]

[16p, 60s] Public Dashboards

- **[12p, 31s] Developer profiles (people as first-class Bugzilla entities)**
  Includes: patch-writer reputation from Code review, trustworthiness from Severity.

  SQ: A means to learn more (e.g., their activity, duration with project, reputation, modules/products, roles) about an individual would improve assignment, triage, and code review assignments.

  * **P15**: "Need some way to figure out who you are so we can treat each other better."
  * **P6**: "Severity can be misused because anyone can set it."
  * **P14**: "It is completely based on the person whose code is being reviewed and his sense of them."
  * **P8**: "People are not first class objects which makes gauging a contributor difficult."

  Idea: P8: "Can't mention a person's name in a comment and have that link to their profile."
  Idea: P15: "Are they a module owner or a peer?"
  Idea: P17: "Security bugs have a different colour background, maybe something similar for first-time patch authors / bugs?"

- **[12p, 22s] Workload transparency**
  Includes: Reviewer load from Code review, Time estimate from Dashboard, Transparency from status.

  SQ: Knowing what others are working on can enable assigning more relevant tasks to people as well as enabling better load balancing.

  * **P11**: "...he can be more helpful if he can better understand what people are working on and how it fits with [their tasks]."
  * **P3**: "[it would be better to] spread the load on key reviewers."
  * **P17**: "[frequently uses the review queue length] to see who might be quickest"
  * **P6**: "Order things by time estimate and priority."

- **[8p, 12s] Communicating interest**
  Includes: Getting people's attention.

  SQ: Being able to indicate interest in a bug without taking ownership is useful.

  * **P8**: "Assign it to himself and then makes a note that others can take over."

  Idea: public watching list (feels similar to CC though).

[15p, 39s] Private dashboards

- **[12p, 21s] Tracking activity**
  SQ: It is hard to determine what has happened since the last time an issue was examined.

  * **P11**: "Gigantic spreadsheet of bugs he is looking at. It would be useful to know how the bugs have changed since he last looked.
  * **P4**: "Sometimes will write reminders in the comments; this is dangerous because he may not go back."
  * **P3**: "They will let it go a few weeks, if there is not traction they will start to follow up."
  * **P6**: "Review status on the dashboard."

- **[10p, 18s] Watch list**
  Includes: Personal priorities.

  SQ: Developers can only track a limited number of bugs in their heads; it would be nice to be able to watch bugs and sort them by activity date.

  * **P8**: "Would like to have a personal list of bugs without claiming them."
  * **P6**: "Wants to get info based on what has changed since last time I looked at it."
  * **P7**: "Private comments for himself."
  * **P15**: "Need a way for people to set their own priorities for bugs."

A.1.2 Collaborative Filtering [4p, 8s]

[4p, 8s] Highlighting important comments
SQ: It is difficult to determine what comments are good and important.

* **P14**: "Just finding the right comments among dozens of comments is difficult."

* **P7**: "Hard bugs that turn out to be complicated - 72 comments, and it's important, you have to read all that history. Similarly, bugs could benefit from a way to mark which comments are good."

* **P10**: "The biggest thing that would their flow would be to pull out important comments on STR."

Idea: P7: "Strike bad information"
Idea: P11: "Being able to mark something as important fact in the comment field"
Idea: P11: "Would like to see nested conversations, so that you can collapse threads that are not important".

A.1.3 Status [18p, 56s]

[8p, 24s] What is the current status?
Includes: Evolution of a bug
SQ: It is hard to determine the current status of a bug, where the bug is at in the process.

* **P9**: "Very daunting to pickup a bug that you have no idea where it is in the process."

* **P5**: "Because there are so many fields, its not clear where the bug is in the process."

* **P6**: "No way to mark a bug in fine grain status if a bug is waiting on something."

* **P4**: "We can't figure out what state the bug is in."

* **P4**: "Sometimes people will see an issue and start working on it not knowing about if a bug exists or not."

Idea: P9: "Triage needs a way to tag a bug to know where it's at."
Idea: P13: "He wants tools that help to understand the evolution of the bug."
Idea: P1: "Bug progress would be good to track."

[11p, 20s] Next action
SQ: Developers want to be able to set/see the state of a bug - is a bug waiting for an approval? are repro steps missing? where the bug stands in the process?

* **P5**: "Having the ability to add a few next actions would be good."

* **P17**: "Would be good to have some way of having a next action that pings them [reporters]."

* **P16**: "We can usefully annotate every bug with a next action field."

* **P7**: "You look at the bug and you think, who has the ball? What do we do next?"

* **P5**: Wants next action, next action assignee rather than a status.

Idea: P10: "Jesse Ruderman's ideas on next steps strikes a cord on him"
Idea: P17: "Next action field would be very helpful (i.e., on whom the bugs is waiting for next action: reporter, triage, dev, AQ etc.)"
Idea: P3: "Would be good to have interface that allowed you to set up 3 things - next step, review, person"

[6p, 12s] Hand-off
SQ: Handoff of bugs is often forced by following up and giving people nudges.

* **P7**: "Assignment often gets ignored. What really works is giving people little nudges."

* **P15**: "You have to watch a bug in a lifecycle and make sure that the hand off is happening."

* **P15**: "You have to push bugs. There isn't a clean handoff and the bugs will stall."

* **P12**: "They have a bug master. This is an experiment called bug master, the person's main task is to watch bug mail and get more detail, make bug actionable. Nag developers when triage follow up is needed."

* **P7**: "If bugs come up that is important, you work them in. Someone comes and asks you about it, you interrupt your project to go work on that."

Idea: P14: "There has been talk in the past of having some way to flag who the next person is for handling a bug."

A.1.4 E-mail [17p, 45s]

SQ: Email plays a fundamental role in how developers keep apprised of what's going on inside Bugzilla.

[8p, 12s] Overwhelming volume
SQ: Email is a primary way of receiving bugs, as well as communicating the whole bug fixing process.

* **P4**: "Sometimes he will just skip a day because he can't keep up with it".

* **P11**: “Way too much information flow coming at him. Bugzilla folder has 8000+ unread and only can look at it a couple of times per month”.

* **P17**: “The person that reports the bug gets hammered with spam with no way to stop. This causes people to avoid filing the Orange bugs as they knowing it’s going to spam them!”

* **P17**: “Email flow is huge, CC changes cause email...”

Idea: P14: “He would like to be able to watch only incoming bugs coming into a component”.
Idea: P17: “It would be handy to have people that edit a bug to say that this isn’t worth sending out to everyone.”

[11p, 16s] Email is important

* **P16**: “Primary way to see new bugs is bug mail. Workflow is very bug mail focused.”

[2p, 2s] Email is not important

* **P11**: “People contact him through back channel like IRC, send a personal email or talk to him.”

[6p, 9s] Filtering
SQ: Some mail clients (e.g., Gmail) do not have good filtering functions, as a result it’s hard to distinguish important emails due to overwhelming number of emails received daily.

* **P17**: “Bugzilla doesn’t let you control the flow enough, 5000 email in a month and most of it doesn’t relate to his work”.

* **P17**: “Not being able to say that you don’t want email from a specific bug is really annoying, especially if you report something on an Automated test”.

* **P6**: “Now uses Gmail that lets you go through bugs quicker but can’t filter by headers in the emails. He has to remember too much information about why he was sent the information as a result”.

* **P14**: “He would like to see a way to stop receiving email from a bug he filed, no way to do that”.

Idea: P6: “Get a better mail client. No solution that works with Gmail”.
Idea: P15: “He uses the xheader filter, it tells you why it was sent to you”.

[4p, 6s] Formatting

* **P17**: "Email formatting makes it very hard to read when a long text field has changed, Gmail doesn't use fixed width font so table distorted. "

A.2 Supporting Tasks [20p, 700s]

A.2.1 Code Review [20p, 130s]

[6p, 10s] Recommending reviewers
SQ: Being able to get a list of recommended reviewers along with their review loads would make it easier to request approval on another component.

* **P15**: "You can always type in a name but a list will help new people."

* **P17**: "Has been asked to do reviews in the past even though he is not a peer."

Idea: P17: Populate reviewer list with info from the wiki Modules page.
Idea: P8: "Potentially a default queue or fake person for a component where reviewers can come and take reviews from".

[10p, 27s] Patches
SQ: Improving the way Bugzilla handles patches would ease and improve the code review process.

* **P13**: "You could find out if the patch will land cleanly, GitHub does this".

* **P14**: "interdif is unreliable so you can't trust it".

Idea: P4: "Would like to compare the two versions of the patch".
Idea: P14: "Would be good to see which comments are attached to what version of a patch".
Idea: P14/P16: "[Supporting] series of patches would be very good".

[8p, 12s] Importance of review timeliness
SQ: Developers rely on code reviews turning around rapidly.

* **P7**: "Pretty good norm that reviews have high priority".

* **P3**: "He will by the end of the day go through his [email for review requests]".

* **P1**: "Review queue is a top priority, keeps it at zero".

Idea: P17: "Review ETA would be good".

[12p, 32s] States
SQ: Due to possible social frictions, review approval states are used inconsistently/wrong and thus increasing risks and vulnerabilities.

* **P7**: "r+ with nits is common. Fix is usually trivial, or simple, full confidence that people can fix it. This could be a mistake."
* **P8**: "Even with regular contributors, there is a negative social aspect to r-".
* **P2**: "Believes that r+ with nits is a huge risk."
* **P16**: "He will use r-. Different people have different habits. People bring up that r- can be a very negative experience, a rejection of all the work you have done."

[8p, 11s] Process & community
SQ: Review process is sensitive due to its nature of dealing with people's egos. People take criticism better if they know a reviewer who is giving the feedback. Differentiating between feedback vs. review can help with this.

* **P2**: "There is no accountability, reviewers says things are addressed, there is no guarantee that the person fixed the changes or saw the recommendations."
* **P11**: "He thinks that the culture around reviews is a bit unhealthy, how do we get that down from few days to few hours... Understanding the human element."
* **P7**: "Someone on the team said that because they know a review is coming, they tighten up their code."

Idea: P20: "He does like a fact that we could make a difference between feedback and review. "

[5p, 8s] Risk / Reward
SQ: Patch approval requires some estimation on the risks involved and how much time should be spend for a review.

* **P2**: "Patch approval needs risk and severity explanation."
* **P16**: "They want a justification. What's the motivation, Risk vs. Reward."
* **P7**: "he tries to make an estimation on the risk when deciding on how much time to spend."

Idea: P3: "Risk analysis should be in the bug when put up for approval. Bugzilla should enforce that if possible, at least suggest it."

[17p, 30s] Misc
SQ: Issues related to the code review process including approval queues, lack of a good system, review response time, etc.

* **P10**: "The review system doesn't seem to be tightly integrated into Bugzilla."

* **P13**: "You could have the test results so that if any fail, the review is automatically rejected."

A.2.2 Triage & Sorting [20p, 259s]

[15p, 35s] Sorting/filtering
SQ: Bugzilla does not provide good sorting and filtering mechanisms and thus these tasks require lot of effort.

* **P13**: "They have a custom tool that doesn't have good sorting capabilities."

* **P9**: "The lack of ability to filter mixed with the volume makes it an overwhelming task. A lot of rework in sorting bugs rather than actually triaging it and moving it along."

Idea: P5: "if users could sort based on tag values, Bugzilla admins wouldn't have to add fields frequently, and Bugzilla's interface wouldn't become cluttered with rarely-used fields."
Idea: P18: "In triage they use a bucket approach, they get through 10 bugs in half an hour. They want to be fair about what bugs they look at. Would be nice to have some form of random sort."

[12p, 25s] Bug assignment:

- **[7p, 10s] Who gets the bug?**

  SQ: Developers usually pick bugs up as they see them. If they do not feel competent, they will assign a bug to someone with the right expertise to fix it. Some bugs are assigned to the community members who are interested.

  * **P16**: "When you connect a bug to the right person who is familiar with it, it gets fixed quickly."
  * **P6**: "If there is a critical bug he will either do it or assign it to someone better suited."
  * **P17**: "Will set assignee to the patch author."

- **[6p, 10s] Self Assignment**

  SQ: Self assignment is pretty common and used as an indicator to others that someone is working on it. Developer who files a bug is often the one who writes a patch for this bug.

  * **P6**: "Sometimes he will create a bug and assign it to himself right away to signal to others he will take care of it."
  * **P4**: "When someone starts working on a patch, they often will assign it to themselves."
  * **P1**: "If you have enough drive to file a bug, you likely have the drive to own it."

- **[4p, 5s] Unassigned**

  SQ: It can be hard to figure out who is working on a bug since some bugs do not have clear owners because developers do not set Assignee field.

  * **P15**: "Many people don't assign a bug to themselves. When you attach a patch, it should get assigned to you."
  * **P6**: "Some people never set the assign field. A fixed bug with no Assignee make life hard."
  * **P9**: "Assigned could help there but there are a lot of bugs that go from new to fixed without ever being assigned."
  * **P9**: "Sometimes they forget the assigned field and the bug gets marked fixed without ever having an owner. This last one is pretty common."

  Idea: P15: "It would be nice if the system told that you have to attach a patch to this or review. Bug hygiene is something we should do."

[17p, 41s] Components/products
SQ: Figuring out what product/component a bug should go is very confusing since Mozilla has so many of them listed and there is no good documentation available. Updating product/component field is also hard.

* **P8**: "If he doesn't know the product, it's a pain point to find. A lot of them are overlapping and vague."
* **P6**: "It's not clear why certain bugs go under certain areas."
* **P11**: "The component and the product categories are extremely confusing."
* **P15**: "When you want to move from one product to another, it doesn't let you change the component."

* **P17**: “Sometimes it’s very hard to figure out in which product/component it should go.”

Idea: P17: “The UI doesn’t update when you change a product, component should refresh.”

[3p, 4s] Component owners
SQ: Having component owners might be useful since their operational knowledge can speed up the triage process.

* **P2**: “Many components don’t seem to have clear owners or the buckets are too big to work through.”

Idea: P16: “ideally he would like to have explicitly a group of developers watching components and having ownership of those.”
Idea: P15: “He thinks that we need to find people that care about certain area. if you split them up, you get a benefit from being familiar with what’s coming in and that speeds things up.”

[4p, 7s] Last touched
SQ: Ordering bugs by last touched date allows developers to monitor recent changes.

* **P10**: “When he was more involved to triaging Firefox General, he used Last Touch Date for figuring out when the last time a bug had changed.”

* **P9**: “Currently the most effective way to triage a bug, look at the most recently touched bugs.”

[10p, 16s] Is this bug actionable?
SQ: Developers would like to have mechanisms that trigger an action from the end-users to provide missing information on a bug, e.g., a test case.

* **P7**: “Decent number of bugs that are not actionable. Crash bug that doesn’t have a repro case but is common. Not sure what to do with that as it’s not actionable.”

* **P6**: “Rather than try to make a repro case, he will request one and move one. Often this can lead to bugs getting lost.”

* **P17**: “the frustrating bugs are the bugs that don’t have enough information and you know that it’s very likely that a bug will not be fixable and it’s a losing battle.”

Idea: P14: “What would be good for triage, would be good to have a way to show what is missing.”
Idea: P17: “If the bug is not actionable and there is no follow up, they just need to be closed out.”
Idea: P12: “Ubuntu, they have a life cycle, they automatically delete bugs after a certain amount of time.”

[4p, 5s] Volume of bugs to triage
SQ: Triage is very difficult due to the overwhelming volume of bugs.

* **P1**: "Feels that he is not aware of all the bugs he should be due to lack of visibility. Currently has about 2500 bugs, no one is aware of the full range of issues in that list due to volume."
* **P16**: "As a general rule, they don't look at the bug list because its just too big."
* **P18**: "They have 1000 open bugs. They have difficulty dealing with that much volume."

[4p, 7s] Duplicates
SQ: Dups are created to track old bugs but the cloning process is harder than needed. Filtering dups needs to be automated.

* **P15**: "He will sort by the resolved and look for dups. Lots of dups show that the bug is out there."
* **P2**: "The cloning process is poor, it's not lightweight and requires you to re-define components as well as making it difficult to clear out information that is not required in the new duplicate."
* **P10**: "Dups were especially useful for looping back to old bugs."

Idea: P15: "It would be awesome if you could compare bugs and come up with a list of potential bugs. This would speed up bug dup clearing."

[6p, 9s] Bugs in "General"
SQ: Bugs filed to Firefox General are likely to be missed since no one owns this component.

* **P9**: "There are also bug that are legitimately general, so it makes it difficult."
* **P1**: "Firefox General - bugs filed under this heading will often go under the radar."
* **P9**: "General is a treasure trove and a graveyard at the same time."

[3p, 4s] Bug kill days
SQ: Bug kill days is a new practice to manage overwhelming number of open bugs.

* **P1**: "Currently working to get this number under control via bug kill days. During this process, they did uncover bugs that they should have been aware of, included patch that had sat around for 5 years but went unnoticed."
* **P9**: "Often bug kill days are mainly about closing bugs."

[5p, 7s] Triage & community engagement
SQ: Engaging the community can help to speed up the triage process by filtering the bugs, determining duplicates, prioritizing bugs, etc.

* **P13**: "People on the mailing list tend to find things pretty quickly. It's pretty easy to get people to test Firefox Aurora and Beta. Its harder for Jetpack, driving engagement is an ongoing issue."
* **P7**: "There could be filtering on some of the bugs by non-engineers. Not sure what would happen, it would take some learning and training."
* **P15**: "It [triage] can be easily distributed. They should find ways to engage volunteers. Incremental approach better than letting a bug log build."
* **P16**: "It would help to have people whose job is just to organize triage efforts."

[2p, 2s] Midair collision
SQ: Miscommunication on who is working on what bug can lead to a midair collision.

* **P10**: "Bugzilla does have a problem with multiple people doing something at the same time, mid air collision. It will happen 1-2 times in a triage session of 100 bugs.Usually this is caused by someone not being clear on who would make the update."
* **P6**: "Midair collisions - someone submits before you do. It's too sensitive."

Idea: P6: "Etherpad has solved it [midair collisions], so it can't be that hard to Bugzilla."

[5p, 5s] Triage meetings
SQ: Triage meetings happen on a weekly basis for most teams. People attend them to discuss issues needing attention.

* **P10**: "Each week they go through all the new bugs."
* **P11**: "He will go to the operational level meetings where e.g. triage is done to talk about it."
* **P8**: "He will attend triage if there are particularly important issues he is working on that are seeking approval."

[2p, 4s] Triage of other components
SQ: Triaging other components is hard since each component has its own policies on the triage process.

* **P9**: "It's hard to triage other groups. Different components have different rules that the developers want triagers to follow."

Idea: P9: "What could help is a guide per component on who they want it done. If that was publicly available, this could help."

[10p, 18s] Triage process
SQ: Current triage process is not manageable as it's very time consuming, yet developers have heavy workloads.

* **P7**: "Spends about a couple hours a week, needs more time than that, but too many other things to do. He does want to change the bug triage process so we can reduce the workload on the engineers."

* **P6**: "This largely causes to be no systematic triage. Doing triage in a systematic way won't get him anything unless it saves time in future."

Idea: P17: "Much of the work in triage could be better if we just asked the right questions up front (e.g., have you tried safe mode, what is the crash ID, etc.)"

[4p, 4s] Comments
SQ: Any activity on a bug is communicated through comments. However, it can be quite time consuming to read through a big list of comments.

* **P8**: "Comments, they work pretty well. There is a strong culture, people have adapted to make comments work well."

* **P4**: "Reading through 30 comments is time consuming."

Idea: Refer to Collaborative Filtering.

[15p, 39s] Misc

A.2.3 Reporting [20p, 145s]

This bucket concentrated on challenges surrounding reporting. Triage-related issues are separated into their own bucket (Triage); while some metadata is here if it is really a reporting issue, a whole separate metadata bucket exists as well. *Includes*: Intimidating for new users from Bugzilla.

[17p, 33s] Submission is harder/more confusing than needed
SQ: Submission process can be hard due to the overwhelming number of fields, non-user-friendly form layout, unknown component/product structure, targeting two groups of users - end-users and experienced developers.

* **P6**: "He thinks it's too much, only should have the summary field and description, possibility to add attachments. Anything else is too much for newcomers."
* **P5**: "Needs to be way faster. Entering a big could be fewer steps."
* **P17**: "He often feels that you never know if a field is important or not. It wastes a lot of time for people to fill in a lot of garbage data when you don't know if anyone is using it."

Idea: P12: "Redesigning the usability, placement and layout would help, not removing things."
Idea: P4: "What shows up on the form biases people to do or not to do certain things. Teaching people what a bug system is, and what information is needed."
Idea: P13: "How the eye is drawn to the more important fields could be improved by changing the layout."
Idea: P4: "It might be better for people to choose between more complex and simple version, getting people to file better bugs."

[10p, 14s] Improving reporting
SQ: Filing bugs can be improved by informing users on how to get their crash data, providing repro steps, etc.

* **P5**: "is attaching a test case and a stack trace, the process of doing one at a time is frustrating. Would be better to do both things at once."
* **P14**: "Would be nice to get a stand-alone website that shows the bug."
* **P6**: "The things that really bugs him about filing a bug - go through product, then go through component. Doesn't always know where to put it."

[4p, 9s] Defects in the reporting experience
SQ: User reporting experience can be improved by fixing issues with bug submission form.

* **P5**: "If the CC is about to be submitted in error, it should tell you that right away, not after you submit it."
* **P15**: "Why is the crash data not filling in the bug? This should be a tool suite."

* **P10**: "The back button duplicates would be nice to fix. People can create multiple bugs by hitting the back button, you often see this as a chain of the same bug submitted several times in a row."

* **P15**: "URL field. Minor thing but frustrating, it often will add http:// and it will double it if you paste an URL into it. When you select the field it should highlight the HTTP."

[10p, 23s] Metadata
SQ: Some metadata fields are more important than others, some are irrelevant. Removing redundant and useless fields can help to improve reporting.

* **P17**: "OS is being set wrong could cause problems for people verifying a bug."

* **P13**: "Some things that cause friction. As a developer he knows that platform should be all."

* **P4**: "Sometimes there are a lot of fields that don't seem needed but every now and then will be needed, like platform, often changes them from a specific platform to all platforms."

Idea: P13: "In past version he has seen bug submission tools that will import info about the environment that you are working on when you report a bug. It would be nice to have this kind of Add bug command."
Idea: P17: "It would be really good to get people together to figure out which fields are necessary to them and why."

[9p, 14s] The role of the description and summary fields
SQ: The summary and description are most important fields and thus should be placed on top of the page. The summary field needs to be space limited since developers would like to see a good short summary.

* **P4**: "People will write very general summaries that lead other people that have different issues to assume it's related and start posting conflicting information. Often too broad - Firefox is too slow, Firefox doesn't work right, Firefox uses too much memory, Firefox stopped responding, Firefox crashed."

* **P10**: "There is a big blank box problem with the summary. Some people will use 2000 words into the summary."

* **P7**: "Gets annoyed when he has to go through bugs that are random thoughts."

* **P15**: "Summarizing the bug first seems backwards, so he puts the description field on top of the summary. "

Idea: P11: "Being able to modify the original description is important."
Idea: P13: "He would like to rearrange the fields so that the summary and the description are higher on page. "

[12p, 15s] Possible enhancements to improve reporting
SQ: Developers provide a number of suggestions on how to improve Bugzilla.

* **P9**: "Ideal would be to have friendly UI for end users, very basic version. Harvest bugs out of that system and put them into Bugzilla."
* **P16**: "We could definitely do a better job at filtering incoming reports before they hit the Bugzilla database."
* **P7**: "He has a template that helps fill in information for js bugs. It fills in the component and a few other fields automatically."
* **P1**: "It would be useful to have a primary assignee and a list of secondary assignees."
* **P2**: "CC is so overloaded it doesn't tell you why you are there. Could mean you wrote the bug, you edited the bug, you wants you to be aware of it, someone commented, someone voting by cc on a bug, needs to be teased out a bit more. "
* **P15**: "Auto-component is useful."

[2p, 3s] External tools
SQ: Bugzilla should not be used as a user feedback tool.

* **P9**: "In terms of end user submission, Bugzilla is not the right tool."

Idea: P2: "input.mozilla.org better for feedback, keeping clutter out of Bugzilla."
Idea: P9: "Input is a good way to help with this."

[7p, 8s] Intimidating for new users
SQ: Bugzilla is quite intimidating to new users. End users often treat Bugzilla as a feedback submission tool.

* **P4**: "A lot of people see it as a complaint form rather than an issue tracking system."
* **P19**: "New people don't know where to file bugs, it should be easier to figure that out."
* **P17**: "He has so many hacks around issues that he forgets how it would be for a new user."

Idea: P9: "Bugzilla, in the long run, should stop being the end user bug tool."
Idea: P15: "We need to make it pretty, we should include Java script."

[8p, 26s] Misc
SQ: Developers are talking about what and how many bugs they receive/file, or why and when they file certain bugs.

* **P7**: "Sometimes people file bugs to talk about ideas. Some optimization we could do to a feature or a similar suggestion."
* **P7**: "Sometimes people will send an email rather than a bug."
* **P4**: "He uses the submit process in a few different ways. Sometimes filing things that he heard from developers in other context, web developers don't always want to deal with the bug system, may be a comment from a mailing list. "

– –

A.2.4 Search [14p, 53s]

[4p, 7s] Quick search
SQ: Quick search is not being used much. It is used for searching through whiteboards, keywords or flags.

* **P6**: "There is a quick search, this helps you with the complexity."
* **P5**: "Quick search, needs a lot of info to get good results."

[5p, 5s] Advanced search
SQ: Advanced search is used by most developers as it's more flexible on what you can search by.

* **P5**: "Advanced search, which is what most developers use."
* **P10**: "He uses the advanced search almost exclusively."
* **P16**: "Some people like quick search, he likes the advanced version."

[5p, 8s] Saved/shared
SQ: Developers want to be able to save and share their search lists, as well as the sort order with others.

* **P10**: "Would be nice to be able to push it to them and have them accept the lists, specifically shared saved searches."
* **P13**: "He would like to have saved searches, he can't save the sort order."
* **P18**: "Provide a link that you can share the same random sorted list. "

[7p, 10s] Hard/confusing
SQ: It's hard and confusing to query Bugzilla.

* **P19**: "When you look at bug lists all day and have to do searches in Bugzilla, it's a pain."
* **P9**: "Running searches on Bugzilla s kind of scary sometimes."
* **P6**: "Querying in Bugzilla is hard. He has to spend a few minutes to figure out how to do the query."

Idea: P19: "It should be really easy to search in Bugzilla, is there a way to do Goggle version of a Bugzilla search?"

[3p, 3s] Date range
SQ: Date range is used to track changes on bugs and to improve search performance.

* **P10**: "He uses the time frame to narrow the scope of search results."
* **P15**: "He will put -7d in the search field and only include bugs that have changed in the last week."

[3p, 5s] Performance
SQ: Search is very slow in Bugzilla.

* **P6**: "Bugzilla is too slow, this is wasting a lot of time, very frustrating."
* **P15**: "If a bug list is too long in the URL, then search fails. It has a problem with a volume in performance. "

Idea: P10: "Restricting the range shortens the query time".
Idea: P15: "We should let Google index the whole thing."

[2p, 2s] Product
SQ: Bugzilla doesn't let you search against several products/components.

* **P10**: "He finds that he wants to search against multiple products. When last he checked, Quick Search couldn't handle that."

[3p, 5s] Dups
SQ: Search is often used to detect duplicates.

* **P15**: "Simple search is good for clustering duplicates. He will limit search to two or three duplicates."
* **P10**: "Summary search is good to avoid duplicates."
* **P5**: "Search to make sure we don't make dups."

[6p, 8s] Misc
SQ: Search is limited by what fields you can search by.

* **P5**: "Can't reliably find bugs to help with."
* **P9**: "Should be able to search by point in triage and keyword."
* **P6**: "No good way to query certain information."
* **P15**: "A whole other category of search to find groups of bugs".

A.2.5 Testing and Regression [14p, 37s]

[5p, 9s] Regression
SQ: Knowing whether and where the bug has regressed is important for tracking bug fixes. If something is a regression of their patch, they will fix it or find the person who broke it.

* **P5**: "If it's a regression will CC the person that broke it."
* **P8**: "Always trying to keep an eye if this bug regressed on a release train. Figuring out where the bug regressed. Making sure that fixes have tracking."
* **P17**: "MozRegression does bisection of nightly downloads. You input a date range and it downloads the nightly builds, it gives you a change set range that you can drop into the bug."

Idea: P7: "If Bugzilla knew it was a regression, then it could help you find where it was. There is just enough data to figure it out, but very manual process in figuring it out."

[8p, 15s] Testing and its reliability
SQ: Complete testing is infeasible. Developers rely heavily on automated testing, more experts are needed to write manual tests.

* **P8**: "We don't have the infrastructure to reliably test things."
* **P15**: "It has to pass the checking test, Thinderbox pass."
* **P12**: "They do a mix of Test Driven Development and writing the test after the fact."
* **P13**: "They have a big test suite that they run through everything. Most of it is automated, they had a hot fix last week that could have been avoided if there was more manual testing."

Idea: P14: "There is very little attempt to broaden the tests. You can't do this without understanding the code well. The way to fix this is to have someone that knows something about what should be going on and can do some exploratory testing."

[6p, 9s] QA & Validity
SQ: Bug verification process is inconsistent. QA involvement varies from team to team. Some QAs do, others do not double check every bug.

* **P15**: "Verification should be eliminated unless it has a clear visibility."
* **P17**: "Most of the bug he has worked on don't need QA verification, since code cleanup/deadcode removal/build system work."
* **P8**: "They will mark a bug as verified once they have checked it. Not every QA team will do that but it's very useful."

Idea: P14: "QA metrics should not be the number of bugs verified, bugs found might be good."

[2p, 4s] Fuzzing
SQ: Fuzz testing bugs are typically fixed right away.

* **P7**: "Fuzz bugs come in, that get fixed right away, sometimes they hang around and there is a risk there that it could be something important."
* **P5**: "Fuzzers are a form of randomized testing."
* **P7**: "Fuzz bugs are important, the source of the bug tells you a lot."

A.2.6 Tasks [13p, 76s]

[4p, 7s] Role-specific views
SQ: Role-specific views (e.g., triage, developer, review) would make it easier to accomplish common tasks for that view.

* **P17**: "Different Bugzilla view modes for different use cases".
* **P10**: Default/New user, QA/Triage, Developer, Review.

[10p, 27s] Release management
SQ: While release schedule is known ahead of time, release drivers are challenged by the uncertainty of what bug fixes are going into next release.

* **P11**: "Bugzilla is terrible for release management... Release manager - it's a burnout job, no one could have it for more than a few months."
* **P12**: "They are on a train cycle: Release branch, stabilization branch, development branch, same 6 week cycle."

* **P11**: "What actually in a release is very hard to comply in Bugzilla."
* **P18**: "Their code development is not directly tied into the train model."

Idea: P13: "It would be nice if there was a large scale version of what fixes are going into the upcoming release."
Idea: P18: "They have a wiki page for every release where they keep notes and will add links to it to maintain history."

[10p, 22s] Where a bug lands?
SQ: It's hard to figure out what version a bug lands. Setting up target milestone can be confusing, error-prone.

* **P13**: "We currently see when it was supposed to land. By looking at it a year later, you have to figure out if that's actually where it landed."
* **P2**: "Doesn't tell you if it landed on Beta or Aurora."
* **P2**: "There is no way to be sure that the version on the bug is correct. A lot of room for failure, no checks and balances."
* **P8**: "Bugzilla only has one resolution which is closed. We need to know what version it is fixed on. Resolved, fixed on this version. "
* **P17**: "Target milestone is set as the landing field rather than the target which is a bit confusing to newcomers."
* **P19**: "There might be uncertainty as to whether to land a fix or not for a branch, and it is because of the differences in opinion about the risk involved."

Idea: P17: "Why not just call it "Fixed for: ""
Idea: P17: "Security team not using target milestone and instead using the status-firefoxN:fixed flags."
Idea: P18: "They are working on a project that will help land patches from Bugzilla [Build:Autoland]"

[8p, 12s] Statistics
SQ: Being able to flexibly create high-level reports (e.g., lists of custom statistics) could create views that must be manually created through searches.

* **P5**: "Statistics would be good; how are things progressing on a larger scale?"
* **P11**: "JS is a good team... they want a graph that is going down."
* **P18**: "It would be interesting to know what the churn rate is on a daily and weekly basis."
* **P19**: "It would be interesting to know the ratio of bugs marked as fixed vs. open."

[6p, 8s] Workflow

* **P11**: "Big flaw is that workflow is not covered by on an individual level."
* **P14**: "His personal workflow is that he writes a patch and then exports it."

A.3 Expressiveness [19p, 188s]

A.3.1 Metadata [20p, 132s]

Includes: Status flags from Status

[14p, 38s] Tracking flags
SQ: Tracking flags are used to raise awareness for a specific bug.

* **P10**: "Flagging is used to raise bugs to Dev and PM teams."
* **P3**: "The flag indicates to engineering that it needs to be looked at."
* **P2**: "Tracking is meant to make sure the bugs don't get lost."
* **P15**: "The tracking bug is a way to make sure it will be looked at several times a week."

[17p, 36s] Whiteboard
SQ: Whiteboards are used a lot during the process management including tracking status, seeking approval, highlighting important information on a bug, marking the version number, or setting a new keyword.

* **P3**: " Use the whiteboards for next step and who will do it, tracking problem areas."
* **P3**: "Using whiteboard when using approval or review."
* **P4**: "Highlight comments that are important to read before commenting," "free-form tagging".
* **P5**: "Whiteboard to order bugs impact."
* **P12**: "They now use whiteboard for triage follow up."
* **P1**: "Using whiteboard flags to manage the process."
* **P18**: "Public shaming is the key way to maintain the whiteboard flags."

Idea: They are free-form and thus hard to search on.

[14p, 22s] Keywords
SQ: Keywords add additional workflow information.

* **P15/P1**: "Keywords as calls for action."
* **P4**: "Keywords are sort of like a tagging system but more restricted."
* **P11**: " Keywords is a substitute for workflow."
* **P2**: "Component owners should be able to set up keywords specific to their area."

Idea: Lightweight keyword system; Christian's keyword system.

[3p, 6s] Meta bugs
SQ: Meta bugs are used for tracking related issues and dependencies among bugs.

* **P5**: "To find bugs again, he uses meta bugs. Has filed 59 meta bugs. Finds them useful for grouping."
* **P1**: "Meta bugs - a master bug per feature, no work goes on in this bug, it tracks all other bug activities going on via dependencies."

Idea: P18: "It will auto create a meta bug for you if it's missing and do a lot of connections for you."

[8p, 12s] Metadata (general)
SQ: There are too many fields to be filled in when reporting a bug, dealing with meta data can be quite time consuming and intimidating.

* **P6**: "The amount of info we need from users is too much."
* **P4**: "Sometimes what scares them off is that there are all sorts of fields they didn't know what to do with."
* **P5**: "Platform means too many things."

Idea: P4: "We need better metadata, but we don't want people to have to enter meta data because they won't do it."
Idea: P17: "Do we even need some of the data he is updating, we should not have to do some of this work by hand. A lot of meta data work for the sake of it."

[3p, 4s] Tagging (general)
SQ: Tagging can eliminate other metadata fields like OS, platform.

* **P5**: "Tagging could get rid of a lot of fields."
* **P5**: "Tagging as a cross b/w the keyword field and whiteboard field."

Idea: A good tagging system.

[9p, 14s] Status flags
SQ: Status flags are misleading and not well understood by developers.

* **P15**: "He doesn't know what "New" in Bugzilla means."
* **P4**: "What does verified state means?"
* **P1**: "Status is only reliably to say fixed or not."

Idea: P4: "It might be better to have a better way to convey clarity of the bug. Unconfirmed, new, assigned are limited. A few things we want to convey: we don't know what you are talking about; we get what you mean, can't repro; we get what you mean, it's not sure it's a bug; we get what you mean, but we are quite enough detail; and many more."

A.3.2 Severity/Prioritization [19p, 56s]

[6p,8s] Unclear definition of priority/severity
SQ: Priority and severity are not well defined.

* **P11**: "Everyone doesn't use the priority levels consistently."
* **P8**: "Priority categories are not well defined... No clear definitions."
* **P5**: "Priority and Severity are too vague to be useful."

Idea: P4: "Needs a 10 word phrase rather than a number 1 to 6."

[13p, 14s] Priority
SQ: Teams can set up their own priority fields. Priority levels are not used consistently.

* **P19**: "Priority is not used at all. People might add P1 if something is really burning."
* **P1**: "Security bugs are always top priority."
* **P20**: "Generally P1 is a blocker, P2 is pretty important. P3 is important, P5 is when you get to it."
* **P12**: "P1 is need, P2 is we want, P3 is nice to have."
* **P8**: "They are using priority field to help organize the large volume."

Idea: P3: "Use tracking for priority. Tracking +".

[11p, 16s] Severity
SQ: Severity s rarely used, mostly to highlight common crash bugs.

* **P5**: "Severity is not used, just something people argue about."
* **P8**: "Severity if rarely used, a few bugs he will set to crash and/or blocker."
* **P6**: "It's usually very evident from the bug if it's severe. "

Idea: P3: "Severity is not used. They used the tracking system. It's either + - or cleared."

[8p, 13s] Prioritization
SQ: Individuals, as well as teams prioritize bugs based on their importance.

* **P1**: "Priorities are heavily influenced by quarterly goals."
* **P5**: "You don't want to prioritize en mass."
* **P5**: "Priority is useful for prioritizing an aspect. Aspect of quality, security, MemShrink, stability."
* **P1**: "Often bugs are ordered on a wiki page during team meetings. Team decides priority as a group."
* **P6**: "He would go after the bigger issues first."

[5p, 5s] Misc
SQ: There is a sense that using priority and severity can be beneficial if there was a well defined system.

* **P5**: "Priority should only come in on large projects."
* **P6**: "He would like to see a priority system, but he doesn't have a good way to do that now."

A.4 The Rest [20p, 117s]

A.4.1 Version Control [8p, 43s]

Issues related to interacting with version control systems, branching, merging, etc.

[3p, 12s] Reviewing via Github
SQ: Some code reviewing is done via GitHub not Bugzilla.

* **P13**: "GitHub lets you do line by line comments. This means you can have a discussion around issues right in the code on each issue very fast. This leads to better fixes, by the time you are done with the review process on GitHub, you likely to have better code."
* **P13**: "They use GitHub more these days. Github has more systems to fill in comments on code."
* **P12**: "Conversations happen in Bugzilla and then in Github once the development starts, so you lose part of the conversation."
* **P12**: "It's hard to use Github mixed with Bugzilla. Sometimes there is development that isn't filed in Bugzilla."

Idea: P13: "It would be good if you could point at specific bits of codes, revisions, delta."
Idea: P12: "He thinks that there is a desire to integrate Bugzilla and Github."
Idea: P12: "The Bugzilla Github add-on is a very useful hack that helps you reference between the two. "

[6p, 13s] Checkins
SQ: Bugzilla doesn't have a mechanism of checking in code.

* **P2**: "No version control on patches."
* **P17**: "If they changed the commit message header after r+ and then re-upload, the process of checking it is manual, the interdiff is really bad, it lies."
* **P15**: "Check in from Bugzilla possible? It could submit the bug number and summary to the HG. HG logs have a lot of typos."

Idea: P13: "Github allows you to get a link to the actual revisions and see how the bug was actually fixed."

[4p, 6s] Branching
SQ: A set of patches normally forms a new branch.

* **P16**: "They will push a branch to Bugzilla as a set of patches."
* **P13**: "When they release a new version, they move trunk to a stabilization branch."

[2p, 4s] Merging
SQ: It's hard to do merging since it requires update of many metadata fields.

* **P17**: "When doing merging, it's a huge job to update all the bug fields. He would merge 20 to 100 changesets per merge, typically 100+ a day."
* **P8**: "He does merges, it's hard to do, because there is no bulk way to change many bugs at once. "

Idea: P17: "Getting this tool [BZexport, BZimport] fixed would be a very good step, since saves having to set assignee on ma bugs after merging inbound. "

[4p, 8s] Integration of Bugzilla with Hg
SQ: There is no direct linkage between Bugzilla and Hg and thus hard to track source code changes.

* **P10**: "Bugzilla doesn't let you dig down, it doesn't give you the changes that happened in a HG range."
* **P13**: "There isn't much integration between Bugzilla and the code they are working with."
* **P20**: "To get richer interaction between the bug and the repository."

Idea: P13: "Bugzilla would need deeper integration with the source control tool."

A.4.2 Bugzilla Issues [16p, 64s]

This is a high-level bucket for random Bugzilla issues that didnt really fit in the other categories.

[8p, 11s] UI
SQ: Bugzilla's UI is not user-friendly or designed for process management.

* **P2**: "Bugzilla is not setup to triage, UI not setup for it."
* **P12**: "Design, usability improvements would make Bugzilla give more information."
* **P15**: "The Bugzilla interface is bad, too many fields."

Idea: P5: "Would prefer to replace Bugzilla. Not a particular software package in mind. Fast, Ajaxy, something that doesn't suck."
Idea: P6: "If Bugzilla stopped working on new features and focused on the big existing problems, he would be a happy person."

[7p, 11s] Advanced scripts and tweaks
SQ: Developers use their own scripts or other 3d party tools to help them through the process and workflow.

* **P2**: "Own tools, Python scripts for common tasks. They build tools to work around limitations using the REST API."
* **P17**: "He has seen people with grease monkey scripts on RedHat Bugzilla/addons doing same thing."
* **P8**: "3d party tools, he uses the Bugzilla Tweaks."

[3p, 3s] Process
SQ: The lack of well structured and defined process makes developers' work harder.

* **P2**: "Process is good, tooling bad."
* **P4**: "I think the biggest issue with going through the bugs is that we don't go through the process at all."

[8p, 11s] Feature pages
SQ: The purpose of having feature pages is not well understood, thus they are rarely used and out of date.

* **P11**: "The feature pages are such high overhead they are not always up to date."
* **P3**: "Feature pages, they have been awful. They are out of date the moment they are finished editing. The biggest issue is that there has been poor communication on what they are for."
* **P6**: "He lost track of what was happening on the development of feature pages. Not sure if they are the best way to track that information."

Idea: P15: "Feature pages, need links from a bug from a blog post, then the bug should be aware of that."

[2p, 3s] Performance
SQ: Performance wise, Bugzilla is slow.

* **P14**: "The speed of Bugzilla is the major issue."
* **P18**: "Bugzilla is slow, takes too long to load a page."

[4p, 6s] Culture
SQ: Since Mozilla heavily relies on community contributions, it's important to provide transparency on how Mozilla operates and establish a culture where communication and education of the community is a priority.

* **P15**: "People are not first class citizens in Bugzilla, bugs are."
* **P4**: "If contributors don't really know how Mozilla works, it can be hard to find out how to edit a bug."

Idea: P14: "If you don't need to have one person do the work, you can have a person that attaches a test case, another one that minimizes it, a third one that finds a regression range, and then a fourth one that fixes the bug. We should encourage people to make a bug take any step forward is good. Community members are often the key resource for doing this work."

[12p, 19s] Misc

A.4.3 Useless [9p, 10s]

These are quotes that really didn't have any specific value. (e.g., "I like turtles").