Situational Awareness: Personalizing Issue Tracking Systems

Olga Baysal, Reid Holmes, and Michael W. Godfrey
Software Architecture Group (SWAG)
David R. Cheriton School of Computer Science, University of Waterloo, Canada
{obaysal, rtholmes, migod}@cs.uwaterloo.ca

***Abstract*—Issue tracking systems play a central role in ongoing software development; they are used by developers to support collaborative bug fixing and the implementation of new features, but they are also used by other stakeholders including managers, QA, and end-users for tasks such as project management, communication and discussion, code reviews, and history tracking. Most such systems are designed around the central metaphor of the "issue" (bug, defect, ticket, feature, etc.), yet increasingly this model seems ill fitted to the practical needs of growing software projects; for example, our analysis of interviews with 20 Mozilla developers who use Bugzilla heavily revealed that developers face challenges maintaining a global understanding of the issues they are involved with, and that they desire improved support for situational awareness that is difficult to achieve with current issue management systems.**

In this paper we motivate the need for personalized issue tracking that is centered around the information needs of individual developers together with improved logistical support for the tasks they perform. We also describe an initial approach to implement such a system — extending Bugzilla — that enhances a developer's situational awareness of their working context by providing views that are tailored to specific tasks they frequently perform; we are actively improving this prototype with input from Mozilla developers.

I. Introduction

Issue tracking systems are used by most software projects. They enable developers, QA, managers, and users to submit bug reports and feature requests and are also often used for other tasks such as bug fixing and code review. Because these systems are also used for project management tasks, there is often some level of integration with development and maintenance activities, and their supporting tools. Most issue trackers are designed around the central metaphor of the "issue" (bug, defect, ticket, feature, etc.) For example, in Bugzilla reported issues have IDs that are easily searched but finding what has changed since the last time an issue was examined is very challenging (one Mozilla developer (P11) maintains a "gigantic spreadsheet of bugs he is looking at. It would be useful to know how the bugs have changed since he last looked")[1]. As a result, developers have poor collective understanding of the overall project status and activities of others within the shared software project [2]–[4]. Our own qualitative study on the developers' experience with the Bugzilla issue tracking system also indicated that developers often have difficulty maintaining a global understanding of the issues they are involved with, and that they desire improved support for situational awareness [1].

Situational awareness is a term from cognitive psychology; it refers to a state of mind where a person is aware of the elements of their immediate environment, has an understanding as to their greater meaning, and can anticipate (or plan to change) the status of these elements in the near future [5]. The term is used in engineering, for example, to describe how air traffic controllers work; it is also an apt description of how software developers must maintain awareness of what is happening on their project, be able to manage a constant flow of information as issues evolve, and be able to plan appropriate actions. Developers often find themselves trying to identify the status of a bug — 1) What is the issue waiting on? 2) Who is working on what bug? 3) What are the workloads of others? 4) 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, or follow up on?.

One way that developers can cope with the challenge of being aware of what is happening on the project is by making issue tracking systems more developer centric. Our work aims to help software developers improve their awareness on the status and recent changes on their issues (as one Mozilla developer (P7) put it, "you look at the bug and you think, who has the ball? What do we do next?") and increase their ability to monitor personal tasks ("What he really wants is a dashboard to show the important stuff he needs to see: review status, assign bug with a new comment or a change since the last time he looked at" (P17), "It would be nice to be able to create a simple list for yourself" (P11)) and to manage information flow ("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)).

By treating people as first-class entities within issue management systems, developers can be provided with a personalized landing page that can have public and private information and serve as a heads up display showing them their own tailored view of the issue tracking system. However, this is not a common approach in these systems, where only issues are first class entities; people are usually modelled as mere meta-data on various issue fields.

In this paper we propose a developer-centric solution organized around personalized landing pages that reduces the

[1]All the quotes are taken from the set of interviews available at https://wiki.mozilla.org/Bugzilla_Anthropology; the results of the qualitative analysis are reported in [1].

issue-centricity of Bugzilla by equipping developers with information filtering and change tracking capabilities. Bugzilla is a widely deployed bug tracking system used by thousands of organizations including open-source projects such as Mozilla, the Linux kernel, and Eclipse, as well as NASA, Facebook, and Yahoo!.[2] Our approach is evolutionary; we augment Bugzilla with the personalized views on the system targeted to help developers gain and maintain ongoing awareness on the project, as well as track progress on their own activities.

II. Related Work

Issue tracking systems have been previously studied by the research community. However, previous work has mainly focused on understanding how issue management systems are being used in practice [6], [7] and suggesting a number of design improvements for developing future issue tracking tools [7]–[10].

Several tools have been developed to enhance developer awareness and understanding of the large source code bases [11]–[15] but none of them target issue tracking systems. Existing research also offers a number of tools [16]–[19] to assist developers with daily tasks and development activities; these tools are more relevant to our research goals. Hipikat [16] provides assistance to new developers on the project by recommending relevant artifacts (source code, bug reports, emails) for a given task. Bridge [18] is a tool that enables a full-text search across multiple data sources including source code, SCM repositories, bug report, feature request, etc. Mylyn [17] is a task management tool for Eclipse that integrates various repositories such as GitHub, Bugzilla, JIRA, etc. It offers a task-focused interface to developers to ease activities such as searching, navigation, multitasking, planning and sharing expertise. Yoohoo [19] monitors changes across many different projects and notifies a developer of any changes in the depend-upon projects that are likely to impact the developer's code.

The state-of-the-art research identifies key design improvements for developing next generation of issue management systems: role-based views, prioritized to-do lists, privacy preservation vs. better transparency, ownership vs. expressed interest. Ideally, new defect management systems would overcome existing shortcomings and combine all desired features in one design. However, bug and issue tracking systems are often implemented as a part of integrated project management systems and are unlikely to be replaced. Thus, our work proposes an initial solution that addresses existing shortcomings and augments a widely deployed bug tracking system — Bugzilla — with developer-centric views to improve developer's global understanding of the project's status and changes.

III. Developer-Centric Approach

To understand how developers currently interact with the Bugzilla issue management system, we performed a grounded theory study to explore the prevailing themes about their engagement with Bugzilla, and to identify the key challenges they face in practice.

A. Open Coding Approach

We have performed a qualitative study on a set of interviews with 20 Mozilla developers on the topic of their engagement and experience with Bugzilla bug tracking system [1]. We split these 20 interviews into over 1,200 individual quotes and performed an open coding approach [20] to gain insight into high-level themes about Bugzilla, with the aim of identifying 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. Table I presents an overview of the concept categories, as well as the count of the participants, quotes, and sub-themes.

TABLE I
Overview of concept categories.

Category	# Participants	# Quotes	# Sub-themes
Situational awareness	19	208	14
Supporting tasks	20	700	53
Expressiveness	20	188	12
(Everything else)	20	166	12

Of the major topics that emerged from our analysis of the interviews, we were most surprised that improved support for situational awareness seemed so important to Mozilla developers; for example, developers expressed frustration that they were not able to easily determine which developer is working on which bug fix at any given moment.

The Mozilla project already makes use of team-wide dashboards to track some aspects of the development effort, e.g., bug open/close charts, performance deltas, etc.[3] In the interviews, developers expressed a keen desire for dashboard-like features that would support personalized list-based views of development. For example, some developers wanted to be able to privately mark certain bugs to track; others wished to be able to explore the role of other developers (Are they active? Which modules do they own? Do they have review rights?); and several developers expressed interest in better transparency in modelling the workloads of others, to aid in the allocation of reviewers for proposed patches.

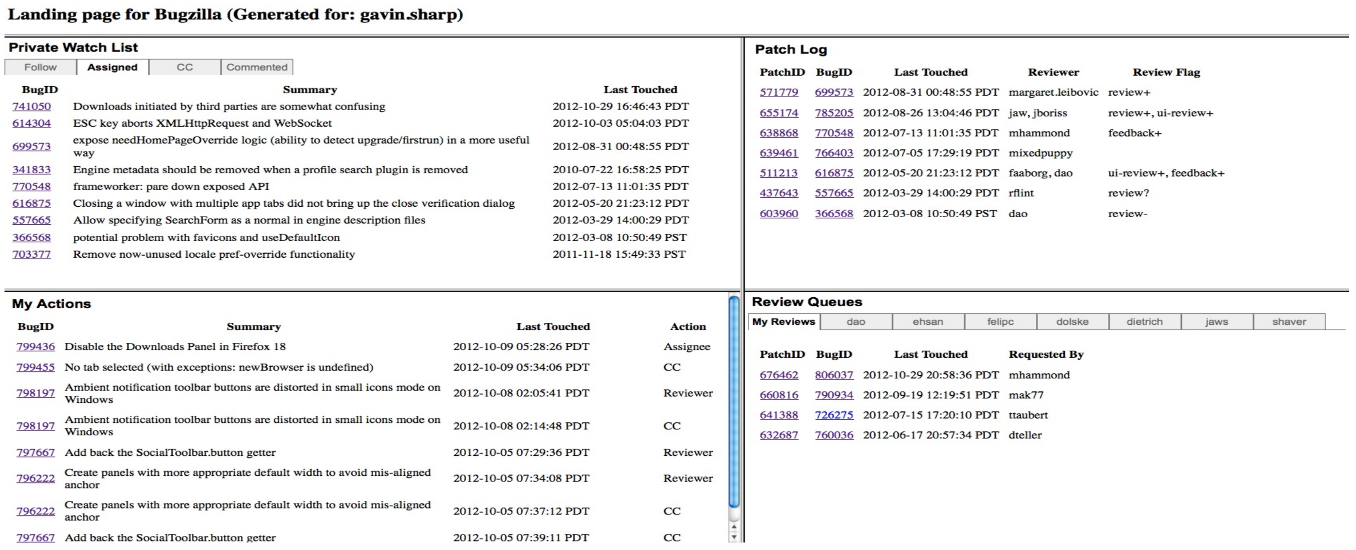
Unlike chart-based views that are geared towards project managers, most of the perceived shortcomings with respect to situational awareness in Bugzilla involve accessing and correlating various pieces of metadata that are already present in the Bugzilla data store, but are not easily accessible by its users.

B. Task-Oriented Views

Based on the results of the qualitative analysis, we identified several ways in which the needs of Bugzilla users could be better met, largely by easing access to key information that already exists within the system but can be hard to obtain.

[2]http://www.bugzilla.org/installation-list/

[3]http://eaves.ca/2011/04/07/developing-community-management-metrics-and-tools-for-mozilla/


Landing page for Bugzilla (Generated for: gavin.sharp)

Private Watch List

BugID	Summary	Last Touched
741050	Downloads initiated by third parties are somewhat confusing	2012-10-29 16:46:43 PDT
614304	ESC key aborts XMLHttpRequest and WebSocket	2012-10-03 05:04:03 PDT
699573	expose needHomePageOverride logic (ability to detect upgrade/firstrun) in a more useful way	2012-08-31 00:48:55 PDT
341833	Engine metadata should be removed when a profile search plugin is removed	2010-07-22 16:58:25 PDT
770548	frameworker: pare down exposed API	2012-07-13 11:01:35 PDT
616875	Closing a window with multiple app tabs did not bring up the close verification dialog	2012-05-20 21:23:12 PDT
557665	Allow specifying SearchForm as a normal in engine description files	2012-03-29 14:00:29 PDT
366568	potential problem with favicons and useDefaultIcon	2012-03-08 10:50:49 PST
703377	Remove now-unused locale pref-override functionality	2011-11-18 15:49:33 PST

Patch Log

PatchID	BugID	Last Touched	Reviewer	Review Flag
571779	699573	2012-08-31 00:48:55 PDT	margaret.leibovic	review+
655174	785205	2012-08-26 13:04:46 PDT	jaw, jboriss	review+, ui-review+
638868	770548	2012-07-13 11:01:35 PDT	mhammond	feedback+
639461	766403	2012-07-05 17:29:19 PDT	mixedpuppy	
511213	616875	2012-05-20 21:23:12 PDT	faaborg, dao	ui-review+, feedback+
437643	557665	2012-03-29 14:00:29 PDT	rflint	review?
603960	366568	2012-03-08 10:50:49 PST	dao	review-

My Actions

BugID	Summary	Last Touched	Action
799436	Disable the Downloads Panel in Firefox 18	2012-10-09 05:28:26 PDT	Assignee
799455	No tab selected (with exceptions: newBrowser is undefined)	2012-10-09 05:34:06 PDT	CC
798197	Ambient notification toolbar buttons are distorted in small icons mode on Windows	2012-10-08 02:05:41 PDT	Reviewer
798197	Ambient notification toolbar buttons are distorted in small icons mode on Windows	2012-10-08 02:14:48 PDT	CC
797667	Add back the SocialToolbar.button getter	2012-10-05 07:29:36 PDT	Reviewer
796222	Create panels with more appropriate default width to avoid mis-aligned anchor	2012-10-05 07:34:08 PDT	Reviewer
796222	Create panels with more appropriate default width to avoid mis-aligned anchor	2012-10-05 07:37:12 PDT	CC
797667	Add back the SocialToolbar.button getter	2012-10-05 07:39:11 PDT	CC

Review Queues

PatchID	BugID	Last Touched	Requested By
676462	806037	2012-10-29 20:58:36 PDT	mhammond
660816	790934	2012-09-19 12:19:51 PDT	mak77
641388	726275	2012-07-15 17:20:10 PDT	ttaubert
632687	760036	2012-06-17 20:57:34 PDT	dteller

Fig. 1. Landing page: developer-centric presentation

We identified four key tasks that developers perform daily: bug tracking, patch tracking, self-tracking, and assigning patch reviewers. We now describe each task and how it is supported by our prototype (shown in Figure 1).

1) **Bug tracking**
For developers who use Bugzilla, email is the key communication mechanism for discussing bugs and the bug fixing process. Any change on an issue results in an email being sent to the developers whose names are on the issue's CC list. For many developers, these emails are the primary means for maintaining awareness in Bugzilla. Developers receive an email every time they submit an issue, edit it, want to be aware of it, someone comments or votes on a bug. An individual developer can track only a limited number of bugs in their head; 10 of the developers who were interviewed wanted to be able to watch bugs and sort them by activity date. One said, "[I] would like to have a personal list of bugs without claiming them" (P8).
Our prototype supports developers with a *Watch List* for indicating their interest on a bug without taking ownership. Watch lists provide means to track bugs privately by adding them to their private watch list without developers having to CC themselves on the bugs. Bugs are ordered by "last touched" time as "last touched time a key metric for tracking if work is being done on a bug" (P1). Watch lists enhance "last touch" with short descriptions and are useful for setting up personal priorities on bugs.
2) **Patch tracking**
Bug fixing tasks are centered around making patches — code modifications that fix the code. While working on a bug fix, developers will often split a single conceptual fix into multiple patches. "People are moving to having multiple patches rather than one large patch. This really helps with the review. Bugzilla isn't really setup for this model" (P16). Ten developers expressed a desire to improve the way Bugzilla handles patches,"It would be good if [Bugzilla] could tell you that your patch is stale" (P13). Developers mainly expressed desire for tracking their own patch activity, as well as determining what patches are awaiting reviews or who is blocking their reviews.
The *Patch Log* view displays developers' patches sorted by last touched date to be aware of the recent changes, as well as indicating the current status of the patch (e.g., the name of the reviewer the patch is waiting on).
3) **Self tracking**
We found that developers face challenges in determining what has happened since the last time an issue was examined (as noted by 12 participants). Some developers mentioned, "[I want] to get info based on what has changed since last time I looked at it" (P6), "You look at the bug and you think, who has the ball? What do we do next?" (P7). They also need some means to observe and track their tasks, for example their review queues. "He has a query that shows all his open review queue" (P16), "The review queues are very useful, he will check that every few days just to double check he didn't miss an email" (P8).
The *My Actions* view supports monitoring developers' tasks including bugs reported and assigned to them, patches submitted for reviews, and discussions on bugs or patches (recent comments).
4) **Assigning patch reviewers**
While 12 participants indicated that Bugzilla is ill suited for conducting code review ("The review system doesn't seem to be tightly integrated into Bugzilla" (P10)), the common task developers perform is determining who

is the “right” reviewer to request a review from. The right person to send a patch for review to may either be the one having faster review turnaround or having a shorter review queue. In order to answer this question, developers need to be informed about reviewers’ work loads and average response time. “I can be more helpful if I can better understand what people are working on and how it fits with [their tasks]” (P11).

Supporting this task is particularly important if a developer is not familiar with the module/component reviewers — “When submitting a patch for another component, it’s more difficult, he has to try to figure out who is good in that component, look up their review info” (P8).

The *Review Queues* view displays developer’s current review queue, provides better transparency on the work-loads of others, in particular it presents review queues of the reviewers to support better decision-making on to whom a developer should ship their patch.

We have implemented a prototype in the form of a personalized landing page that provides developers with private watch lists, patch logs, and action summaries that can help Bugzilla users have better awareness of the issues they are working on, other issues of interest to them, as well as tasks they perform daily. Our solution is organized around custom views of the Bugzilla repository supporting ongoing situational awareness on what is happening on the project. Since community members report defects to Bugzilla, the developer is presented with the means to select bugs of interest to them. Some UI components are injected directly into Bugzilla, while the main solution is based on filtering important and relevant information from the repository and presenting it to the developers supporting their common tasks such as bug fixing, feature implementation, code review, triage, etc. Our prototype is built into the existing issue tracking system targeted to enhance Bugzilla with the means to increase developer’s awareness within the situational context.

IV. Conclusions and Future Work

Our qualitative study of interviews with Mozilla developers suggests that they often have difficulty maintaining a global understanding of the issues they are involved with, and that they desire improved support for situational awareness. We proposed an initial solution that improves support for particular tasks individual developers need to perform by presenting them with the custom views of the information stored in the issue management system. By personalizing issue management systems, developers can stay informed about the changes on the project, track their daily tasks and activities.

Our prototype has received positive feedback from Mozilla developers and is actively being extended with input from them. Once the prototype is fully implemented, we plan to conduct a user study with Mozilla developers to evaluate its effectiveness in an actual development setting. Such a study would reveal whether our prototype can reduce developers’ efforts in achieving and maintaining awareness on their issues of concern and how they evolve. We also plan to seek developers’ feedback on the prototype and its perceived value during the project management process.

References

[1] O. Baysal and R. Holmes, “A Qualitative Study of Mozillas Process Management Practices,” David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada, Tech. Rep. CS-2012-10, June 2012. [Online]. Available: http://www.cs.uwaterloo.ca/research/tr/2012/CS-2012-10.pdf

[2] J. T. Biehl, M. Czerwinski, G. Smith, and G. G. Robertson, “Fastdash: a visual dashboard for fostering awareness in software teams,” in *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*, 2007, pp. 1313–1322.

[3] R. E. Kraut and L. A. Streeter, “Coordination in software development,” *Commun. ACM*, vol. 38, no. 3, pp. 69–81, Mar. 1995.

[4] D. E. Perry, N. Staudenmayer, and L. G. Votta, “People, organizations, and process improvement,” *IEEE Softw.*, vol. 11, no. 4, pp. 36–45, Jul. 1994.

[5] M. R. Endsley, “Toward a theory of situation awareness in dynamic systems: Situation awareness,” *Human factors*, vol. 37, no. 1, pp. 32–64, 1995.

[6] N. Bettenburg, S. Just, A. Schröter, C. Weiss, R. Premraj, and T. Zimmermann, “What makes a good bug report?” in *Proc. of the ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2008, pp. 308–318.

[7] D. Bertram, A. Voida, S. Greenberg, and R. Walker, “Communication, collaboration, and bugs: the social nature of issue tracking in small, collocated teams,” in *Proc. of the ACM conference on Computer SupportedCcooperative Work*, 2010, pp. 291–300.

[8] T. Zimmermann, R. Premraj, J. Sillito, and S. Breu, “Improving bug tracking systems,” in *proc. of the International Conference on Software Engineering - Companion Volume*, may 2009, pp. 247 –250.

[9] S. Just, R. Premraj, and T. Zimmermann, “Towards the next generation of bug tracking systems,” in *IEEE Symposium on Visual Languages and Human-Centric Computing*, sept. 2008, pp. 82 –85.

[10] S. Breu, R. Premraj, J. Sillito, and T. Zimmermann, “Information needs in bug reports: improving cooperation between developers and users,” in *Proc. of the ACM conference on Computer Supported Cooperative Work*, 2010, pp. 301–310.

[11] R. DeLine, M. Czerwinski, and G. Robertson, “Easing program comprehension by sharing navigation data,” in *Proc. of the IEEE Symposium on Visual Languages and Human-Centric Computing*, 2005, pp. 241–248.

[12] R. DeLine, M. Czerwinski, B. Meyers, G. Venolia, S. Drucker, and G. Robertson, “Code thumbnails: Using spatial memory to navigate source code,” in *Proc. of the Visual Languages and Human-Centric Computing*, 2006, pp. 11–18.

[13] J. Froehlich and P. Dourish, “Unifying artifacts and activities in a visual tool for distributed software development teams,” in *Proc. of the International Conference on Software Engineering*, 2004, pp. 387–396.

[14] L.-T. Cheng, S. Hupfer, S. Ross, and J. Patterson, “Jazzing up eclipse with collaborative tools,” in *Proc. of the 2003 OOPSLA Workshop on Eclipse Technology eXchange*, 2003, pp. 45–49.

[15] A. Sarma, Z. Noroozi, and A. van der Hoek, “Palantir: raising awareness among configuration management workspaces,” in *Proc. of the International Conference on Software Engineering*, 2003, pp. 444–454.

[16] D. Čubranić and G. C. Murphy, “Hipikat: recommending pertinent software development artifacts,” in *Proc. of the International Conference on Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2003, pp. 408–418.

[17] M. Kersten and G. C. Murphy, “Using task context to improve programmer productivity,” in *Proc. of the ACM SIGSOFT international Symposium on Foundations of Software Engineering*, 2006, pp. 1–11.

[18] G. Venolia, “Textual allusions to artifacts in software-related repositories,” in *Proc. of the international Workshop on Mining Software Repositories*, 2006, pp. 151–154.

[19] R. Holmes and R. J. Walker, “Customized awareness: recommending relevant external change events,” in *Proc. of the IEEE International Conference on Software Engineering*, 2010, pp. 465–474.

[20] M. Miles and A. Huberman, *Qualitative Data Analysis: An Expanded Sourcebook*. SAGE Publications, 1994.