An Exploratory Study of Awareness Interests about Software Modifications

Miryung Kim
Electrical and Computer Engineering
The University of Texas at Austin
miryung@ece.utexas.edu

ABSTRACT

As software engineers collaboratively develop software, they need to often *analyze past and present program modifications* implemented by other developers. While several techniques focus on tool support for investigating past and present software modifications, do these techniques indeed address developers' awareness interests that are important to them? We conducted an initial focus group study and a web survey to understand in which task contexts and how often particular types of awareness-interests arise. Our preliminary study results indicate that developers have *daily* information needs about code changes that affect or interfere with their code, yet it is extremely challenging for them to identify *relevant* events out of a large number of change-events.

Categories and Subject Descriptors

D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement

General Terms

Human Factors

Keywords

Empirical study, code change, awareness interests

1. INTRODUCTION

Collaboration is essential to building and evolving large-scale software. Software engineers typically spend 70% of their time working with others and team activities account for 85% of the costs of large software systems [9, 12]. At the same time, software evolution—continual correction, adaptation, enhancement, and extension of a software system—is inevitable; over 90% of the cost of a typical software system is incurred during the maintenance and evolution phase. Moreover, as much as 80% of a developer's time is spent on knowledge discovery in evolutionary phases [14]

There are a number of tools that can help developers investigate code change history and provide awareness about other developers' software modifications. Hipikat [6] helps developers to discover historical context and rationale when they need to implement changes similar to past software modification. Palantir [17] and FASTDash [2] help developers know about other developers' modifications that may lead to merge conflicts. Jazz [3] and CollabVS [10] allow developers to set watch-points such as "notify me when Alice is done with the class `ShapeBounds`" to monitor about software modifications made by other developers. While all of these tools intend to help developers know about other developers' modifications, do these tools indeed address developers' awareness interests? What types of awareness interests do developers have about other developers' code changes? How often does a particular awareness interest arise?

In order to gain a deeper understanding about developers' awareness needs about software modifications, we conducted a focus group study with five professional software engineers. Furthermore, we compiled various types of awareness-interests based on a literature survey, and conducted a web survey with software developers to understand how often developers have a certain kind of awareness needs.

The study found that *different stakeholders* often reason about software modifications *at a different abstraction level* and that *users' awareness-interests are rapidly-evolving* as their tasks change. The users are left to *filter out irrelevant code modifications* such as the changes that do not semantically affect their own changes or to *ignore insignificant changes* such renaming or indentation changes from a large volume of check-in notifications. Currently, this filtering process requires substantial effort for developers to identify and analyze software modifications relevant to their tasks, focus, and interests. These findings suggest the need for a more flexible tool that developers can select and customize the notion of *relevance* and easily search and monitor relevant code modifications.

2. RELATED WORK

Workspace awareness systems such as FASTDash [2] and CollabVS [10] provide information about other developers' task status or activities. Palantir [17] can assist programmers in detecting structural conflicts early by monitoring changes in other programmers' workspace in real-time. Nightwatch [15] and CVS-Watch monitor other developers' activities using programmer-specified watch-points. YooHoo [11] mitigates this problem in some degree by filtering out changes according to predefined rules; however, it is limited to API declaration changes that lead to build errors.

Ko et al. [13] studied software engineers' information needs by observing their daily activities and analyzing observation logs. Sillito and Murphy [18] studied the types of questions that programmers ask to assess how well existing tools support those questions. As opposed to general software engineering tasks such as program comprehension, bug finding, and expertise location, our study focuses on awareness-interests about software modifications.

de Souza et al. [8] studied change impact management approaches in two different organizations. Our study complements this study by providing how often developers have a certain types of awareness interests about other developers' changes. Costa et al. [5] studied the scale, range, and volatility of coordination requirements using the version history of five projects. While their study infers coordination needs based on version histories, our study investigates when and how often developers have awareness interests about other developers' changes using a focus group and a survey.

3. A FOCUS GROUP STUDY ON CODE CHANGE INVESTIGATION

To understand software developers' awareness interests about code changes, we conducted a focus group study with professional software developers enrolled in the Option III masters' program at UT Austin. Based on an earlier screening questionnaire which gathered professional demographic information including their job title, programming languages, years of software development experience, and their comfort levels with various existing software engineering tools (e.g., *diff*, version control, bug tracking, etc.), we recruited a group of participants who frequently use *diff*-like tools, version control systems, and reviewed the modifications of other developers in a peer review setting.[1] All participants had used bug tracking software and investigated using code change histories using a version control system interface such as *SVN log*. Table 1 summarizes the participant profile.

We led the focus group through a brief demonstration of software tools that can be used to investigate, search, and monitor past and present software modifications. We audio-taped the discussion and transcribed the recorded conversation. Key findings from the study and supporting quotes are organized by the questions raised in the study.

In which tasks contexts, do you need to know about others' code modifications?.

Developers need to know about program modifications made by other developers, because (1) others' program modifications may affect their code, (2) it is their job duty to know about others' code and peer review their changes, and (3) they must be able to take over others' responsibility when colleagues are unavailable—e.g., they are sick or leave the company.

"we are required know about code changes because if someone can't do it, you are supposed to have the capability to do it."

"...it obviously affects your code. You only have a part of it, but each part affects the other parts of it."

[1]One participant did not provide their name on the original survey, so we do not have their exact information.

What types of code modifications done by other developers do you want to know about?.

Change notifications without content are useless for developers. They want to know about changes made by other developers that may **depend on** or **interfere** with their own code.

"I work for a small company, and having somebody tell me there was a change is almost useless, because I work with 3 other developers—if they make a change, I've heard them make the change."

"I want to know what other developers are doing, because their code changes will *impact* my code."

"I want to know when interfaces change, you know. Especially again going back to the *dependence structure* of the application, I want to know when those *external interfaces* change."

Developers often want to know about the *evolutionary context* in which the individual changes came about.

"The program has an evolution, but each of the changes themselves has an evolution."

"So you want to see the *history of evolution*, for how this thing came to the point where it's at. Because of the constant need to enhance, you want to see what happened before."

Software modification information should be tailored to each stakeholder's needs (developers, team leads, architects, non-technical managers, document writers, and testers).

"And he is the CTO of that company, so he's technically savvy, and even though he doesn't like seeing our code check-ins, he does like seeing high level changes to the code."

"We use open source too. We're mostly interested if there were bug fixes that we have to worry about."

What are limitations of using existing tools to know about others' modifications?.

Developers noted that information overload and a lack of relevance make current change notification mechanisms inadequate, and potentially harmful in terms of productivity.

"Once a week I get a big laundry list. It's gotten to the point that I see the email and I delete it..."

"I think the change has to be related to me, to my community, to my project."

"So now I get this [SVN notification email], all these files changed... it changed the import statement and I don't care about that."

Information should be pulled only when users request it, or must be aggregated into a single unobtrusive feed. Push-based notifications such as emails and pop-ups could distract users.

"(Email notifications are) not productive. It just stops you, and you gotta look at it. You gotta decide whether it is bad or useless."

"As a programmer I'd like to minimize interruptions as much as possible. Because it just takes you away, especially interruptions that are work related."

"So if I am going to have some kind of notification I probably would like a feed, not an email."

Users should have the ability to control and customize information that they want to know about.

"Are you allowed to see this release information about products about to go out? In my company that would be a big issue."

"So you got a customer who wants to see why the changes

Years	Job title	Languages	Do you regularly use the following tools or conduct the following tasks?				
			diff	version control	bug tracking	peer code reviews	change history investigation
25	Software Analyst	Python (10yrs)	Yes	CVS, SCCS, VSS	Bits	Yes	No
15	System Analyst 3	C++ (4yrs), Java (9yrs)	Yes	Microsoft VSS, SVN	No	Yes	Yes
8	Team Lead	C#, Java	Yes	SVN, Vault, Surround	FogBugz	Yes	Yes
2.5	Software Engineer	Java (5yrs), Javascript (3yrs)	Yes	SVN, CVS	Bugzilla, Jira	Yes	Yes
11	Software Architect	Java (7yrs), SQL (7yrs), XML (10yrs)	Yes	ADE	Bugzilla	Yes	Yes

Table 1: Focus group participant profile

happened, you got a developer who wants to see not only why, but how, and what, so there's a lot more details there."

"So in the case of my company where they need to know what everyone is doing, as a developer I'm not engaged in the every day detail of the project, but I want to have an overview—it's his project, he's working on it."

The results show that developers want to know about code changes made by other developers, but discovering relevant change events requires tremendous effort. The developers also would like to know about the context in which code changes are introduced and they want to know about changes at a different abstraction level depending on their roles. The results motivate a tool that enables flexible means of monitoring only relevant software modifications and investigating the evolutionary context of code modifications.

4. AWARENESS NEEDS ABOUT CHANGES

To produce a prioritized list of awareness-interests that are important to developers, we conducted a web survey. We first compiled a list of awareness interests about software modifications based on a literature survey. Table 2 lists them along with citations to the articles that inspire each awareness interest statement.

By sending a study announcement email to about 60 UT software engineering graduate students and professional software engineers, we recruited 25 study participants. The participants had 2 to 20 years of software engineering experience (8.36 years on average), and they consist of a diverse group of professionals including 17 developers, 3 managers, 2 system analysts, 3 graduate research assistants with prior industry experience, etc. We then asked the participants to indicate how often they agree with each statement: Daily (D), Weekly (W), Monthly (M), Seldom (S), and Never (N).

Table 2 shows our preliminary results with 25 participants sorted by a frequency score: Score = 4|D|+3|W|+2|M|+|S|. It includes the number of responses for each option. The rightmost column is the computed score. The sorted list shows that the impact and rationales of code changes are one of the most frequently sought awareness information. In fact, code change interference, impact, and dependence are three of the top five information needs, and more than a half of developers have *daily* information needs about the impact of and interference with other developer's modification.

5. STUDY LIMITATIONS

Though multiple focus groups are usually recommended to calibrate the results of focus groups, we have conducted only a single focus group so far. For the web survey, all questions were selected by the authors. Therefore, the prioritized list of questions may be missing questions that participants would find more important. While every attempt was made to ensure questions were understandable, participants may have misinterpreted the intent of questions. Due to a small number of participants (5 in the focus group and 25 in the survey), the study results may not generalize to a broader community of software developers. As the survey relies on self-reporting of how frequently developers have a certain type of awareness interest, the results need to be carefully interpreted. Findings in both studies may have been biased by developers' own past development experiences and the organizations that they work for.

6. CONCLUSIONS AND FUTURE WORK

We conducted a focus group study with professional software engineers to understand their awareness needs about software modifications and limitations of existing tools to investigate, search, and monitor code changes. The study found that different stakeholders often reason about software modifications at a different abstraction level and that *users' awareness-interests are rapidly-evolving* as their tasks change. The users are left to *filter out irrelevant code modifications* such as the changes that do not semantically affect their own changes or to *ignore insignificant changes* such renaming or indentation changes from a large volume of check-in notifications. Currently, this filtering process requires substantial effort for developers to identify and analyze software modifications relevant to their tasks, focus, and interests. These results are aligned with the findings from prior work on change impact analysis, awareness, and coordination [8, 17, 5].

As various notions of delta-relationships (e.g., interference, dependence, similarity, and co-occurrence) can be used to identify relevant software modifications, our position is not to re-invent these code change analysis but to provide a flexible analysis framework in which these definitions can be imported, selected, and combined to search and monitor relevant modifications.

Acknowledgment. The author thanks Shayne Czyzewski for his assistance with a focus group and a survey. This work was supported in part by National Science Foundation under grant CCF-1043810.

Statement about awareness interests	**daily**	**weekly**	**monthly**	**seldom**	**never**	**score**
I want to know whose code changes semantically interfere with my recent code changes. [18, 2, 17, 1]	13	4	4	4	0	76
I want to know whether my code is impacted by this change. [18, 2, 17, 1, 13]	11	4	7	3	0	73
I want to know how the use of this API changed. [18]	9	9	2	5	0	72
I want to know whose code changes my code depends on. [1]	13	2	3	6	1	70
I want to know why the code was implemented this way. [13]	9	8	5	3	0	70
I want to know whom I can go to for help with my task. [16, 1]	5	12	4	4	0	68
I want to know the context in which a piece of code appeared. [19, 1]	9	5	6	5	0	68
I want to know the files changed in this revision. [7, 16]	8	7	5	4	1	67
I want to know which coworker of mine has changed this class (method, field, etc.), and when. [7]	7	7	6	4	1	65
I want to know what my coworkers have been doing. [13]	7	7	3	8	0	63
I want to know which bugs were fixed in this revision. [1]	6	6	8	4	1	62
I want to know how the resources (such as APIs) I depend on have changed. [13]	8	4	5	8	0	62
I want to know how development tasks are distributed among my coworkers. [19]	7	6	3	9	0	61
I want to know which parts of the code are unstable. [19]	8	4	7	3	3	61
I want to know in which revision this piece of code was modified. [7]	6	8	3	6	2	60
I want to know which revisions semantically interferes with my recent code changes.	7	5	6	5	2	60
I want to know which revisions my recent code changes depend on.	6	6	5	7	1	59
I want to know who owns this piece of code. [1]	5	6	6	6	2	56
I want to know which files are often changed together. [19]	5	5	5	9	1	54
I want to identify who are the key contributors of this project from this date to this date. [16]	5	3	3	11	3	46

Table 2: Responses to our survey on awareness interests

7. REFERENCES

[1] A. Begel, Y. P. Khoo, and T. Zimmermann. Codebook: discovering and exploiting relationships in software repositories. In *ICSE '10*, pages 125–134, ACM.

[2] J. T. Biehl, M. Czerwinski, G. Smith, and G. G. Robertson. Fastdash: a visual dashboard for fostering awareness in software teams. In *CHI '07*, pages 1313–1322, ACM.

[3] L.-T. Cheng, S. Hupfer, S. Ross, and J. Patterson. Jazzing up eclipse with collaborative tools. In *eclipse '03: Proceedings of the 2003 OOPSLA workshop on eclipse technology eXchange*, pages 45–49, ACM.

[4] O. C. Chesley, X. Ren, and B. G. Ryder. Crisp: A debugging tool for java programs. In *ICSM '05*, pages 401–410, IEEE Computer Society.

[5] J. M. R. Costa, M. Cataldo, and C. de Souza. The scale and evolution of coordination needs in large-scale distributed projects: Implications for the future generation of collaborative tools. In *CHI'11* (to appear)

[6] D. Cubranic and G. C. Murphy. Hipikat: recommending pertinent software development artifacts. In *ICSE '03*, pages 408–418, IEEE Computer Society.

[7] B. de Alwis and G. C. Murphy. Answering conceptual queries with ferret. In *ICSE '08*, pages 21–30, ACM.

[8] C. R. B. de Souza and D. F. Redmiles. An empirical study of software developers' management of dependencies and changes. In *ICSE '08*, pages 241–250, ACM.

[9] T. DeMarco and T. R. Lister. *Peopleware : productive projects and teams / Tom DeMarco & Timothy Lister.* Dorset House Pub. Co., New York, NY :, 1987.

[10] R. Hegde and P. Dewan. Connecting programming environments to support ad-hoc collaboration. In *ASE 2008, 15-19 September 2008, L'Aquila, Italy*, pages 178–187, 2008.

[11] R. Holmes and R. J. Walker. Customized awareness: recommending relevant external change events. In ICSE '10, pages 465–474, ACM.

[12] C. Jones. *Programming productivity / Capers Jones.* McGraw-Hill, New York :, 1986.

[13] A. J. Ko, R. DeLine, and G. Venolia. Information needs in collocated software development teams. In *ICSE'07*, pages 344–353, 2007.

[14] N. H. Madhavji, F. J. C. Ramil, and D. E. Perry. *Software evolution and feedback: theory and practice.* Hoboken, NJ: John Wiley & Sons, 2006.

[15] C. O'Reilly, P. Morrow, and D. Bustard. Improving conflict detection in optimistic concurrency control models. *SCM*, pages 61–69, 2003.

[16] A. Sarma, L. Maccherone, P. Wagstrom, and J. Herbsleb. Tesseract: Interactive visual exploration of socio-technical relationships in software development. In *ICSE '09*, pages 23–33, IEEE Computer Society.

[17] A. Sarma, Z. Noroozi, and A. van der Hoek. Palantír: raising awareness among configuration management workspaces. In *ICSE '03*, pages 444–454, IEEE Computer Society.

[18] J. Sillito, G. C. Murphy, and K. D. Volder. Asking and answering questions during a programming change task. *IEEE TSE*, 34(4):434–451, 2008.

[19] L. Voinea, J. Lukkien, and A. Telea. Visual assessment of software evolution. *Sci. Comput. Program.*, 65:222–248, March 2007.