CVS integration with notification and chat: lightweight team support

ABSTRACT

Communication and awareness have been identified as key issues for effective software development. Code management systems like Concurrent Version System (CVS) can play an important role in this work, but often a

[6, 11]. This might involve current parallel activity or it might involve some unknown past or some unpredictable future activity.

Configuration management systems have been identified as playing a key role as formal tools to support this coordination and awareness. We will overview the specific experiences that have been reported with these systems and also overview more recent visualisation work making use of the data they contain. First however, it is worth reviewing the importance of communication as part of software coordination since communication figures in various ways in the following discussions.

2.1 Communication and Awareness

Communication is talked about in different ways in discussions of software development coordination: as mediated communication and informal communication. Both are critically important for supporting software coordination and complement one another.

Mediated communication happens through information being conveyed or accessible to an individual, often via formal project mechanisms such as CVS logs, project documentation, wall charts, and so on. This has connotations of passive one-way communication regardless of whether that information is communicated automatically or explicitly sought out by the person. Such mediated communication can result in increased awareness of what is going on, which in turn can help a person coordinate their activities with others.

Informal communication on the other hand is a socially-embedded process involving two or more people engaging in ad hoc discussions and interactions. Numerous studies highlight that software engineers spend a significant proportion of their time communicating. Perry et al. [17], for example, found that developers spent 50% of their time in interactive activities other than coding, and that 75 minutes per day was spent in informal communication; De Marco and Lister [5] similarly report that 70% of developer time is spent in communication. Other empirical studies of software teams confirm the prevalence and importance of informal communication and spontaneous ad hoc encounters, e.g., over the water cooler or by dropping into an office [4, 10, 16]. It is in these discussions that ideas are discussed, problems solved, conflicts negotiated, missing information filled in, and so on. In a study of 65 projects/563 individuals, Kraut and Streeter [16] showed that informal discussions with peers was the most highly valued coordination technique and that ‘other people were the most used and valued sources of help’.

It is no wonder then that co-location or at least physical proximity are important factors in the frequency and quality of informal communication [15, 16] and that coordination problems can be exacerbated in distributed development teams because of the increased difficulty of informal communications including barriers such as lack of unplanned contact, knowing who to contact about what, cost of initiating contact, ability to communicate effectively, and lack of trust [12, 13].

However, Kraut and Streeter also point to issues with informal communications. Apart from the need to be local to be most effective, there are the transaction costs of engaging in lots of pairwise interactions and the ephemeral nature of verbal discussions compared to more archival sources such as email or logs.

We now turn to configuration management systems as one of t that the developers actually sent the email before they checked-in code to give “a brief description of the impact that their work [changes] will have on other’s work”. The purpose of this was to give others time to “prepare for and reflect about the effect of their changes”, often resulting in people coming to ask about the change or asking for a delay, etc.

Yamauchi et al. [23] also studied two different distributed open source projects, both using CVS, where mailing lists were pivotal for coordination and awareness. Before check in, developers would extract “the difference between the modified version and the central master code with [the command] ‘diff’ and then submit the differences to a mailing list” (p333).

In all of these cases email serves to meet an informal communication and awareness need, but the informal communication takes place in parallel to, but separated from the CVS logs that anchor the discussions. They also require explicit effort (apart from the automated sending of changes) on the part of the developers to send the initial message and further describe or discuss the changes.

Despite this, most developers reported that they felt that the combination of a CM tool and email worked well. Email provided a way of making [illegible] registered interest. Because the underlying event model in Elvin is very generic, a producer or a consumer can be anything from a software component to a person and it can be put to uses ranging from systems-oriented middleware messaging, to people-oriented filtering of information feeds from source

Figure 2. tickertape conversation triggered by CVS commit message.

In the example presented in figure 2, it is late at night and Phelps is working from home fixing a bug in some code. When he is finished, he checks the file back into the CVS repository, entering the comment “The gap doesn’t actually need to draw anything”. This check-in event causes an Elvin notification to be generated stating the name of the modified file, ‘gap.c’, and the associated comment. The notification is then sent by the server to people subscribed to the ‘elvin’ group. David (‘d’) is working late back in the office and sees that Phelps has made some changes to the ‘gap.c’ file (history line 1). He sends a message joking “and you tell me to go to bed!!!!;-)” (line 2). Phelps and David then engage in some light-hearted banter about their working habits (lines 3-4). Phelps goes on to explain a bit more about what else he has been working on and they have a short discussion around that work (lines 5-7). In the middle of the discussion, David has a problem with his tickertape The (a) Tj 37 0 0 37 336 8 Tm(t) Tj 37 0 0 37 768 0 Tm ( ) Tj 37 0 0 37 360 0 Tm (h) Tj 37 0 0 37 37 01Tm (l) Tj 37 0 0 37 388 0 Tm (l) Tj 3 there were qualitative changes in the types of messages written and, predicting that the increased awareness afforded by tickertape might prompt developers to give more information about code changes, compared the lengths of messages. As developers on the Orbit project did not necessarily work exclusively on that project throughout its course, we did not compare frequencies of commit messages, as any findings could have been attributable to changing work commitments.

In section 6 we go on to compare the findings from the case study to the much larger Elvin projec m

Table 2: Frequency of occurrence of types of messages

	Frequency of messages coded as containing text belonging to the category (percentage of total)	
Category of log entry	Pre Elvin notifications	Post Elvin notifications
Description	168 (58.1)	112 (61.9)
Basic description	52 (17.9)	20 (11.0)*
Effect	68 (23.5)	49 (27.1)
Rationale	55 (19.0)	40 (22.1)
Future/ incomplete	14 (4.8)	8 (4.3)
Value judgement	12 (4.2)	5 (2.8)
Empty message	18 (6.2)	0 (0)**
Invitation	0 (0)	0 (0)
Landmark	2 (0.7)	3 (1.7)
Unsure/ hopeful	1 (0.3)	2 (1.1)
Communication	2 (0.7)	1 (0.6)
Named other developer	12 (4.2)	7 (3.9)
Smiley	1 (0.3)	1 (0.6)

Pre Elvin notifications: N=289; post Elvin notifications: N=181. Pre and post frequencies were compared for each of the categories using the !^{2} *statistic.*
**p<0.05*
***p<0.01*

As table 2 shows, there was a decrease in the number of basic descriptions and empty log messages after CVS messages started to be sent as Elvin notifications. There was no significant change when the cvs2ticker script was implemented, were deleted. 12129 log entries were used in the analysis

6.3 Word Counts

Word counts were calculated for each of the remaining messages in the Elvin CVS log. Counts ranged from 0 words to 119. The mean word count was 9.0 (standard deviation = 8.3)

The correlation of word count with time was calculated to investigate whether the increase in commit message length found in the Orbit log might extend to the much larger Elvin CVS log. A small, highly significant positive correlation was found between word count and time (r=.071, p<.001), a finding in line with the interpretation that the increased public availability of CVS commit messages afforded by tickertape may have encouraged developers to write longer descriptions of code changes.

Correlations of commit message word counts with time for individual developers are tabulated in table 3.

Table 3: Correlations of word count with time for individuals

Develope

The volume of tickertape

[4] Curtis, B., Krasner, H., and Iscoe, N. 1988. A field study of the software design process for large systems. Commun. ACM 31, 11 (Nov. 1988), 1268-1287. _

[5] De Marco, T. and Lister, T. *Peopleware: Productive Projects and Teams. 2nd Ed*. Dorset House Publishing, New York, 1999.

[6] de Souza, C.R.B., Redmiles, D., Dourish, P., 'Breaking the Code', Private and Public Work in Collaborative Software Development. In Proceedings of the *International Conference on Supporting Group Work* (Group 2003) (Sanibel Island, FL, November 2003) 105-114.

[7] Fitzpatrick, G., Kaplan, S., Mansfield, T., David, A., and Segall, B. 2002. Supporting Public Availability and Accessibility with Elvin: Experiences and Reflections. *Comput. Supported Coop. Work* 11, 3 (Nov. 2002), 447-474.

[8] Fi