CVS integration with notification and chat: lightweight team support

Geraldine Fitzpatrick
Interact Lab, Dept of Informatics
University of Sussex
+44 1273 678982
g.a.fitzpatrick@sussex.ac.uk

Paul Marshall
Interact Lab, Dept of Informatics
University of Sussex
+44 1273 678941
p.e.marshall@sussex.ac.uk

Anthony Phillips
Interact Lab, Dept of Informatics
University of Sussex
+44 1273 678983
a.d.phillips@sussex.ac.uk

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 at some time removed from the original entries. The focus of this paper is what happens to a software development team's use of CVS when the log is synchronously augmented with an event notification system, Elvin, and a tickertape tool where CVS messages are displayed and where developers can chat with one another. Data from interviews and a high-level log analysis demonstrate that the tool was effective in supporting timely interaction around CVS entries and became an important communication and awareness tool. Analyses of the CVS logs of two different projects show that, when using the tool, developers tend to include more information in the messages they write when they check code in to the repository.

Keywords

Event notification, CVS, Elvin, awareness, communication, chat, log analysis.

1. INTRODUCTION

Software development is widely recognized as a difficult and complex endeavour, not least because of the involvement of multiple people and the complex interdependencies among the intangible software artefacts being worked on. This can be further complicated by people being distributed across time and space. Effective *coordination*, then, becomes a significant challenge for software development teams and *awareness* and *communication* are regarded as key elements of the way in which coordination is achieved.

It is not surprising then that significant effort has focused on supporting coordination. Much of this support is through explicit project mechanisms such as formal project documentation, the use of formal specification languages, regular structured review meetings, and systems for bug tracking, version control/ configuration management (CM) and so on [2, 12, 13, 16]; these mechanisms are in common use in real-world projects. Grinter [10] clearly identifies the role of CM tools in creating visibility in the development process and becoming a form of organisational memory.

More recent approaches, especially in the research realm, show a move away from managing activities and workflow *per se* to providing visualisations to support awareness of activities and artefacts in software development, e.g., see Storey et al. [21] for a survey of current approaches such as Palantir [18], Jazz [14] and Augur [9]. Many of these visualisations are created by extracting information that already exists in tools, such as version control systems, and representing it in different ways to provide graphical overviews of the code base that can be queried and interacted with. Sarma et al. [18] describe how this transforms the use of version control tools from information pull to information push.

While awareness and communication are both talked about as important aspects of coordination supported by the tools above, communication is treated similarly to awareness as 'information delivery' to the individual. More interactive and discursive communication takes place externally from the above mechanisms, through tools such as email or IM or an informal pairwise basis or at team meetings. Interestingly though, such informal communications have been repeatedly identified as critical for software coordination [4, 5, 10, 12, 13, 16, 17] and there are reports of very effective use of email alongside CVS [6, 11]. The costs of pairwise communications have also been well documented, such as being expensive in time, ephemeral in nature and tightly aligned to physical co-location and accessibility [15].

Hence CM tools play an important role in awareness and passive communication; visualisation tools enhance this by making CM information more temporally relevant and available with additional functionality; more interactive communication however, has to happen separately from these tools.

In this paper, we report on the augmentation of CVS with real-time notification and chat in the same interface. A simple event notification service sends CVS log messages to a tickertape (or other client interface) that also supports informal chat among the development team. The main contribution of this paper is to show how a very simple lightweight event notification mechanism and multi-function client interface, closely integrated with use of an existing tool such as CVS, can support both the awareness and informal communication needs inherent in software development. Data presented from interviews and vignettes, and from analysis of CVS log files, suggest that developers come to regard the entry of CVS log comments more as a communicative act and change their behaviour to provide more information; both awareness and informal communication are supported.

2. BACKGROUND and RELATED WORK

Software development is a complex undertaking, with complex dependencies across code modules and also across time. This creates interesting requirements for coordination and awareness support. Developers report that it is important to know things, such as who is working on what or what changes have been made

[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 the complementary formal communication mechanisms for software coordination.

2.2 Configuration management systems

Configuration management (CM) systems are an important part of how software development teams coordinate and manage multiple software components and multiple people working on those components [10].

Concurrent Version System (CVS) [1] is one type of configuration management system. CVS maintains a current central repository of source code along with multiple previous versions. It offers an optimistic approach to configuration management by supporting parallel development and allowing multiple developers to work on the same code. Developers can take a copy of a current file by checking it out, make changes, then check it back (also termed 'commit') into the repository. Support is provided on check-in for identifying and resolving conflicts. De Souza et al. [6] talk about the importance of this check-in process for transitioning code segments from a private workspace to the public repository.

Associated log files are a part of this transition process in a CVS repository. When developers check code back into the repository, they are prompted to enter a comment into the log about the changes they have just made.

The focus therefore is mainly on the role of the tool in providing a central consistent code base from a more formal, process-oriented view for coordination support [22], e.g. through timeline views, being able to see the status of a checked-out artefact and related code modules, and highlighting (or resolving) differences between versions of the same code [9]. All of these mechanisms require explicit effort on the part of the developers; with many CM systems, this information is only available when a developer goes to check out or check in a file or if they explicitly query the system [18]. Further, these mechanisms do not support more informal discussions and negotiation (articulation work as described by [22]) around the formal process aspects [11, 12, 13].

2.2.1 CM and Mailing Lists

In view of this, many different practices have been developed to augment CM tools so that code changes can be more easily communicated to the wider development group and to enable associated discussions to happen. Mailing lists form an important part of these practices (e.g., see [6, 11])

Gutwin et al. [11] studied three distributed open source projects that used CVS and found that the developer mailing list was both a primary communication channel and awareness mechanism. When a file is checked into the CVS, the change is automatically sent to the mailing list. Just from reading the subject line, developers reported that they could 'keep an eye on' what type of changes were being made and by whom. The developers also used the mailing list for both dissemination and discussion: to send short emails stating what they have done or what they are going to do, and to engage in discussions, e.g. about a bug or a proposed feature.

De Souza et al. [6] describe a software development team of 25 members who made use of email in conjunction with a CM tool to coordinate their work and to move work from their private space back into the 'public' realm. While they had a formal team process guideline about sending an email to the developers' mailing list after changes had been made, de Souza et al. found 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 actions on intangible software artefacts publicly available in a timely way [6, 11]. People became aware of interdependencies they were otherwise unaware of; they could start to get a better sense of who was working on what and what areas of expertise others had; it also served as a learning mechanism [6]. Gutwin et al. [11] highlight the effort-benefit disparity in this parallel use of email with CVS (the people reading the message derive more benefit than the person who has to put in the effort to send the message) but the developers did not mind this as they knew they would also benefit at some other time from someone else's effort. However, even though the developers stated that the combination worked for them, some studies [6, 22] observed situations where it did not, for example rushing to commit changes first to avoid being the person to deal with merges.

Interestingly though, the CVS log itself, which Grinter [10] identifies as an important organisational memory and coordination resource, does not figure directly in these interactions. The extra detail contained in the developer-generated emails is often not associated directly with the log entry, creating a separation of the action on the artefact and the discussion around it, and leaving the burden with the reader to construct the full context for the discussion.

Interestingly too, the CVS log files do not appear to feature prominently for the developers as important mechanisms of coordination, given the reports from various studies of people using CM tools [6, 11]. When accessed explicitly by developers, it is at a time removed from the actions documented in the log. Gutwin et al. [11] also found that while "the commit log is the only awareness source that is based on the actual manipulations of the project artefacts" (p.77), some developers found that it was too time-consuming and tedious to read about numerous commits and to identify the ones of interest to them.

What this all tells us is that developers perceive that they derive sufficient benefits from mailing lists to make the effort, and indeed the avoidance workarounds, worthwhile. However, the benefit is from the timeliness of the emails and discussions not from the CVS logs per se. The system that we will talk about here provides the timeliness of mailing lists for both awareness and stimulating discussions but does not require specific effort on the part of the developer to send out the initial notification nor to process an inbox of email messages.

2.2.2 *CM and Visualisation Systems*

More recent approaches seek to exploit the coordination and awareness role of configuration management systems for developers, but without requiring explicit effort or parallel separated mechanisms such as email. A number of visualisation tools have been developed to provide awareness of activities and artefacts in software development by extracting and manipulating information from existing tools such as CVS [21]. Palantir and Augur are two such examples. Palantir uses an event notification service, Sienna, to collect and distribute relevant workspace information (about the actions of other developers on artefacts) which is then organised and presented via a graphical visualisation on the developer's desktop. The creators of Palantir talk about this as continuous awareness versus the discrete isolated information usually derived from these tools [18, 22]. Similarly, Storey et al. [21] talk of feedthrough awareness.

A common feature of many of these approaches is that they make use of existing tools such as CVS and existing information held in these tools. They do not require additional explicit effort on the part of the developer (unless it is to set up more refined filtering of the information displayed to them) to make that information available. They do however require significant screen real estate if benefit is to be derived from having such continuous views. They are also yet another dedicated application that the developer has to have running on their machine. The approach we will talk about here also makes use of existing information without explicit effort but uses a client interface that takes up minimal screen real estate (similar to the Palantir tickertape interface but in contrast to the graphical visualisations) and that the developers already have running on their machines for a variety of other uses.

Many of the visualisation approaches support awareness through mediated communication, however there is one system, Jazz, that also incorporates a chat facility [14]. Similar to sticky notes [3], Jazz provides a facility for developers to leave chat boxes visibly anchored in the code to support conversations and collaborations in context. By definition then, these will tend to be asynchronous interactions separated in time and requiring explicit effort or serendipity to come across as opposed to the more synchronous and discursive nature of the chats generated by the system in this paper.

3. CVS WITH ELVIN AND TICKERTAPE

The approach we talk about here also makes use of existing tools such as CVS but does so by using information available at the check-in of code to the CVS, in combination with an event notification/router service called Elvin [19, 20] and a multi-purpose client interface such as a tickertape [7, 8, 19, 20]. After introducing Elvin and tickertape and its use with CVS, we will go on to explore how it was used in two different projects and over a period of seven years.

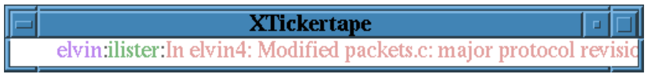
Elvin has been undergoing continuous development during the period of use and data collection that we will be analysing here. At its most general level of description, Elvin provides a means of *content-based addressing*, sending simple structured messages from some producer and allowing consumers or receivers to select messages of interest through use of a subscription [19, 20]. The Elvin server then routes the messages received to those who have 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 sources such as the web, and informal chat.

For the people-oriented uses, a client interface is required to display the messages received and to support the initial setting up of the subscriptions. Again over the extended period of the cases to be discussed here, a number of different client interfaces have been developed but at a general level of description, an interface in common use is a single-line scrolling tickertape that also has the option of being viewed as a threaded chat window.

The users define producers as ‘group’ names and people can subscribe to messages sent to that group, with optional additional content-based filtering to personalise the information they see. There is no technical limit to the number of different subscriptions/groups a user can have via the same client. The same ‘group’ can handle both system-generated messages and user-generated messages as both have the same message structure. When a message is sent to the client, the user will see the name of the group the message is sent to, the name of the person or software that has sent the message, and the content of the message. MIME attachments can also be included with the message.

The tickertape interface is highly configurable and users can control parameters such as the look and feel, the scrolling speed, and the time-out of messages. While the Elvin server itself does not offer any persistent store of messages, users can optionally choose to keep their own log of messages sent to tickertape. A whole-of-groups view can be kept via capturing logs of all messages sent to the different groups; such archival logs have formed the basis of our study here.

Fitzpatrick et al. [7] describe in some detail the variety of uses and experiences with Elvin and tickertape. In this paper, we want to focus on the use of Elvin and tickertape to augment CVS being used by two software development teams. In this case, the developers have set up a group with their project name and CVS has been augmented with a short script called ‘cvs2ticker’ to generate an Elvin message whenever a person checks code into the repository. The message sent to the tickertapes of those subscribed to this group will have the form of <group name, person checking file in, content (file name and log message)> with an automatically included link to the web interface for CVS[1]; this is shown in Figure 1. Developers can choose to use the same group to send a chat message to others, e.g., in response to that CVS-generated message; this would have the format of <group name, person sending the chat, content of chat message>.



elvin:ilister:In elvin4: Modified packets.c: major protocol revisio

Figure 1. Tickertape window displaying CVS message

As was discussed in Section 2.2.1, De Souza et al. [6] talk about CVS commits being an act of moving code from the private to the public realm. However, it is only potentially public because it relies on other people going to look at the log to see the information now available. The use of Elvin-enabled tickertape messages turns a potential for public availability into a much more immediate event that becomes publicly available via targeted ‘broadcast’. This can support implicit awareness in the first instance as other people in the team can glance at the message (cf. Palantir [18]) and support the opportunity for timely informal communication.

In the next section we first describe, using data from tickertape logs and interviews, how CVS-generated tickertape messages came to be used as an important awareness tool by the Elvin development team, and how they integrated into tickertape discussions. Then, motivated by comments made by some of the developers during interviews, we go on to investigate whether making CVS commit messages more temporally available to developers through Elvin/tickertape may have had an impact on the content of those messages.

4. INTEGRATION OF CVS MESSAGES INTO TICKERTAPE CHAT

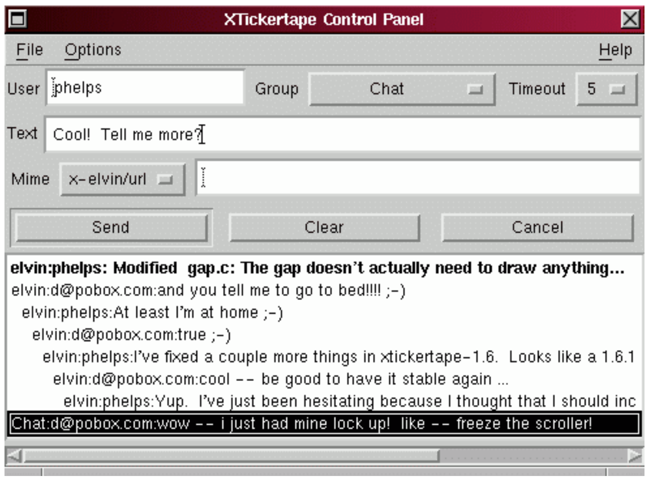
As discussed previously, Elvin was a long running software infrastructure project at DSTC. It resulted in a spin-off company in July 2003 and continues to be developed. Throughout the period October 1993 to December 2004, a total of 15 developers contributed to the project, resulting in 32506 logged changes to the code. CVS was instrumented to send out Elvin notifications to a group called ‘elvin’ in October 1998.

Here we overview the general use by the Elvin project team of Elvin-enabled CVS via tickertape to determine its utility in supporting communication and awareness. This overview is based on an ongoing analysis of a log of tickertape messages. This tickertape log contains 59472 tickertape messages logged between August 1997 and April 1998 and between February 1999 and July 1999. It contains several types of messages sent to several groups, including numerous chat messages, CVS commit messages, news messages and content generated by various bots.

Throughout the tickertape logs there is repeated evidence of tickertape being widely used in conjunction with CVS and becoming an integral part of the way these developers ‘do’ development work. The fact that the tool has remained in voluntary use over seven years and has continued to be evolved is testament to the value the developers derive from using it.

The following is a typical example of how a CVS commit message sent to tickertape triggers a conversation between distributed team members involving friendly banter as well as timely work discussion.

[1] See http://elvin.dstc.com/projects/producers/cvs2web.html for a description of cvs2web. This includes a further link to a cvsweb view of the repository (for a description, see http://www.freebsd.org/projects/cvsweb.html).



User phelps Group Chat Timeout 5
Text Cool! Tell me more?
Mime x-elvin/url

```
elvin:phelps: Modified  gap.c: The gap doesn't actually need to draw anything...
elvin:d@pobox.com:and you tell me to go to bed!!!! ;-)
  elvin:phelps:At least I'm at home ;-)
    elvin:d@pobox.com:true ;-)
      elvin:phelps:I've fixed a couple more things in xtickertape-1.6.  Looks like a 1.6.1
        elvin:d@pobox.com:cool -- be good to have it stable again ...
          elvin:phelps:Yup.  I've just been hesitating because I thought that I should inc
Chat:d@pobox.com:wow -- i just had mine lock up!  like -- freeze the scroller!
```

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 that illustrates the bug Phelps has been trying to fix: "–i just had mine lock up! Like – freeze the scroller!" to which we see Phelps starting to respond in the dialogue text box "Cool! Tell me more?". And so they continued to discuss the new problem.

Table 1 contains a different example showing the use of tickertape to negotiate and schedule inter-dependent work and how the implicit messaging of a CVS notification is reinforced with a follow-up chat message.

Table 1. Tickertape log extract

Time	User*	Message
05:06	alan	Modified quad_ticker.htmi: some reworking
05:07	alan	Brian – when you're done, we'd like to move the edst sub-dir to the public web. Can you ticker when you're ready?
05:08	brian	I need at least 10 more minutes...
05:09	alan	cool ...
05:12	brian	Modified quad_elfs.htmi: (Hopefully) clarified wording.
05:12	brian	Ok, I'm done wordsmithing.

** Names have been changed to preserve anonymity*

These are compelling examples. Often multiple people would participate in the discussions. Even if people didn't directly take part, the others subscribed to the group could oversee what was happening. Interviews with the developer group, both in 1998 and 2005, show experience themes consistent with many of those reported in Section 2.2.1 where mailing lists were used: greater awareness of what was going on and the importance of timely information and timely discussion. One developer in a city over 1000 miles away from the rest of the team stated that it was an "absolutely essential" part of how he was able to work in the team. The developers talked about it changing the way they used the CVS logs, going to the logs much more frequently to get an overview of what was happening either by following a link attached to a message or catching up with the tickertape log at various points, e.g., at the start of the day (especially when other team members work across time zones), and following links from there.

CVS via tickertape integrated with and augmented the developers' everyday work environments to add another layer of information and communication support for both implicit and informal communication. Via one interface, they have been able to find out about computer-based events as they happen, engage in social chit-chat, have a timely work discussion, 'be there' when a problem happens and then engage in collaborative diagnosis, and negotiate the flow of work around a code check-in. They did not have to go to separate tools for notification and for chat. They did not have to forgo their preferred work environments. The implicit communication provided via tickertape happened without any explicit effort on anyone's part. Contribution to a chat around the message was entirely discretionary.

While there is clearly evidence, by virtue of ongoing use alone, that tickertape provides a useful awareness and communication tool for developers, there were two issues that arose from the interviews that made us curious about if and how the use of Elvin/tickertape impacted the content of the CVS logs themselves.

The first was the developers' use of tickertape to promote 'good' software engineering practices. One of the team was known to frequently commit code back to the CVS without a comment. One of the other developers wrote a short 'empty message watcher' script that detected an empty CVS entry and immediately sent a message to tickertape drawing attention to the empty CVS message. On interview, the main culprit said that the messages may have changed his practices but that they were more likely to have changed the practices of others watching (he was the lead developer). We were curious whether there was an effect on the number of null entries as the developers believed.

The second was when one of the developers reported that he felt he changed the content and frequency of his log messages because he wanted to be seen as working hard and was aware of it being more like a communicative act rather than just making an archival log entry that no-one would read.

While the Elvin project CVS logs did not contain enough entries logged prior to the augmentation of CVS to send out tickertape messages to make a valid before and after comparison, we had access to the CVS log of another project that did: the Orbit project.

In the next section we report findings from an in-depth case study of CVS commit messages written by the main developer on the Orbit project. We carried out a content analysis to determine if 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 project CVS log to determine whether trends in that data might be comparable.

5. CASE STUDY OF ORBIT CVS LOG

5.1 Setting

The aim of the Orbit project was to develop a groupware tool. The Orbit CVS log comprised 3122 entries generated between March 1997 and June 2000. Seven developers contributed code to the project. CVS commit messages started being sent as Elvin notifications on the 28th April 1998.

Jack Peterson[2] (JP) worked as the main developer on Orbit between March 1997 and November 1999, making 71% of the changes to the code. He was the only developer to have worked on the project both before and after CVS was augmented to send out Elvin notifications, therefore this analysis focuses on CVS commit messages composed by him.

5.2 Selection of data for analysis

Before analysis, the Orbit CVS log file was processed to remove duplicate commit messages. Duplicate messages occurred frequently when developers checked-in multiple files using the same commit message. A log entry was classed as a duplicate if it contained the same message, was sent by the same user, and was sent within 60 seconds of the one preceding it. Deleting duplicates reduced the size of the log file from 3122 to 1409 entries.

The log file was further reduced in size by deleting all entries recorded during the period from 10th March 1998 to 3rd March, 1999 when Tickertape messages were not being logged; although Elvin notifications started being generated from CVS commits on the 28th April 1998, it was unclear how much information was being included in the generated messages at this time, and they were therefore excluded from the analysis.

Log entries from the first week of the Orbit log were also deleted. This was because they did not represent activity by the development team typical of that which occurred throughout the rest of the project. Finally, all log entries not generated by JP were removed. The processed data used in the analysis comprised 289 log entries recorded before CVS commit messages started to be sent as Elvin notifications, and 181 entries after. These entries were produced by the same developer, and at a similar stage in the development process.

5.3 Coding the content of messages

A coding scheme was developed by selecting approximately 5% of the Elvin CVS log entries and categorising them according to the content of their commit messages. Whenever a message was encountered that couldn't be classified according to an existing category, a new category was defined. The coding scheme comprises messages containing the following categories:

[2] Name changed to preserve anonymity

Description: a description of changes made to the code that named the part of code changed. E.g. "Added event images to the Artifact Avatars, and fixed IconView layout"

Basic description: a description of changes made to the code that did not name the part of code changed. E.g. "Minor modifications"

Effect: effects of the changes made to the code. E.g. "Added the ability to change the title of the VideoView"

Rationale: a rationale for changes made to the code. E.g. "Added db reader/writer code to fix logout problem"

Future/ incomplete: either mentions future work to be carried out, or that changes to the code being checked in have been started, but not completed. E.g. "LocaleViews now can be modified and reverted (Naming yet to come...)"

Value judgment: contains a value judgment about the quality of the code. E.g. "Improved the eloquence of the error messages..."

Empty message: commit message left blank.

Invitation: contains an invitation for other developers to look at or work on the code checked in. E.g. "...feel free to change it"

Landmark: points to the significance of the changes to the code in terms of the project. E.g. 'finally in sync with the changes made the day of "The great disk crash"'

Unsure/ hopeful: expresses uncertainty about the effects of changes to the code. E.g. "Fixed a race condition?"

Communication: contains a statement that is explicitly communicative in intent. E.g. "note the forced NOP for everything"

Named other developer: contains the name of one of the other developers

Smiley: the commit message contains a smiley, E.g. ":-)"

Only three of the categories in the coding scheme are mutually exclusive: a message can only contain a description or a basic description, but not both, and if a message is classed as empty, it obviously cannot contain any of the other categories.

This scheme was then applied to the remaining messages in the Orbit log by the second author; messages were classified as either containing or not containing text from each of the categories. No messages were encountered that could not be classified with the coding scheme.

Reliability was assessed by having the third author apply the coding scheme to a randomly selected 20% of the log. Inter-rater reliability was estimated as high with a Cohen's Kappa of between .81 and 1.0 for each of the codes.

Frequencies of categories of commit messages were compared before and after CVS messages started to be sent to tickertape. Word counts were calculated for each CVS commit message and compared before and after.

5.4 Frequency of log entry categories

The frequencies of CVS commit messages categorised as containing text belonging to each of the thirteen categories in the coding scheme are listed in table 2.

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 χ^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 in the frequency of any of any other categories of message.

5.5 Word Counts

The number of words in CVS commit messages logged before and after CVS commit messages started to be sent as Elvin notifications are summarised in figure 3.

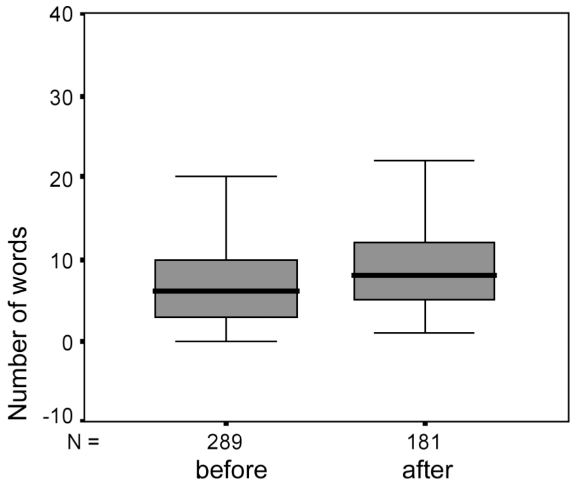


Figure 3: Box plot summarising word count before and after CVS messages started to be send as Elvin notifications.

The mean number of words in a CVS commit message (mean = 9.96, standard deviation = 8.74) sent after CVS had been augmented to send out Elvin notifications was significantly greater than the mean number of words (mean = 7.98, standard deviation = 7.56) in messages logged beforehand (t(468) = -2.599, p=0.005). This wasn't simply due to the reduced number of empty log messages in the later period. When empty log messages were removed from the analysis, the mean number of words was still larger after the cvs2ticker script started being used (t(450) = -1.881, p<0.05)

5.6 Summary of case study findings

We had initially expected to find qualitative changes in the types of commit message written after they started to be broadcast via tickertape, especially as developers had reported at interview that they viewed them as more of a communicative act. However we found there to be few. One possible explanation for this finding is that as CVS messages appear in the same tickertape interface as chat messages, there is no need for the developers to be explicitly communicative in the messages they write; tickertape-enabled CVS messages may play a purely passive awareness function embedded within ongoing chat conversations between developers.

Messages were found to be significantly longer after they started to be sent to tickertape, there were significantly fewer basic messages and the number of empty log messages decreased to zero. We interpret these changes as being in line with developers' comments about the importance of CVS messages on tickertape in providing timely information to the development team; the perception of an increase in the usefulness of information in CVS commit messages influenced JP to give more, and more specific information about his code changes, and to completely stop leaving messages blank.

Although we were unable to carry out a comparison of Elvin CVS log entries before and after they were sent as tickertape notifications, we were keen to test whether effects found in the Orbit log might exist as trends in the larger project and extend over time and across individuals. We therefore selected Elvin CVS messages sent as tickertape notifications and calculated the correlation of both word count and the existence of empty messages with time.

6. Elvin CVS log

6.1 Setting

Elvin was a larger project than Orbit both in terms of timescale and number of people involved in the project; 11 developers worked on Elvin in the period between October 1998 and December 2004, 5 of whom were experienced developers with several years experience of working on software project, and 6 of whom were undergraduate or postgraduate students with varying levels of experience working on software projects.

6.2 Selection of data for analysis

Given 6 years of Elvin CVS log entries numbering 32506 messages, we decided to make a systematic selection of data as a subset for analysis. The first step was to process the log to remove duplicate messages. As before, a log entry was classed as a duplicate if it contained the same message, was sent by the same user, and was sent within 60 seconds of the one preceding it. Removal of duplicates reduced the size of the log to 12564 entries. Finally, log entries that occurred before October 1998, 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

Developer	Description	N	r
1	Developer/researcher	1226	-0.085**
2	Lead developer/researcher	1857	0.026
3	Developer/researcher	17	-0.180
4	Developer/researcher	2	a
5	Developer/researcher	2908	0.026
6	Student	2178	0.183**
7	Student	440	0.274**
8	Student	1189	-0.033
9	Student	380	0.027
10	Student	1197	-0.088**
11	Student	30	0.426*

N=number of messages, r = Pearson's r statistic
**p<0.05, **p<0.01, a=N too small to calculate*

6.4 Empty log entries

Overall, the number of empty log messages was low in the Elvin CVS log: only 112 of the 12129 (0.9%) messages contained no text.

The presence of empty log messages was correlated with time to investigate whether the decrease in empty messages found in the Orbit log might extend to the larger project.

A small, but highly significant negative point biserial correlation was found between the presence of empty log messages and time ($r_{PB}= -.073$, $p<.001$).

Correlations of empty log messages with time for individual developers are tabulated in table 4.

Table 4: Correlations of empty log messages with time for individuals

Developer	Description	N	r_{PB}
1	Developer/researcher	1226	-0.141**
2	Lead developer/researcher	1857	-0.027
3	Developer/researcher	17	a
4	Developer/researcher	2	b
5	Developer/researcher	2908	a
6	Student	2178	a
7	Student	440	-0.186**
8	Student	1189	0.024
9	Student	380	-0.153**
10	Student	1197	-0.114**
11	Student	30	a

r_{PB}=point biserial correlation, N=number of messages
**p<0.05, **p<0.01,*
a=no empty messages, b=N too small to calculate

6.5 Summary of Elvin CVS log

While we were unable to conduct a pre and post tickertape analysis of the Elvin log as we did with Orbit, we can look at trends of use with tickertape-enabled CVS over an extended period of time for a number of people. Overall, these trends were in line with the findings of the Orbit study: the length of messages increased over time and the number of empty messages decreased.

There do seem to be individual differences in relation to the increase in word count: only students showed significant increases in word count over time. It is not possible to determine from the logs whether this was due to the increased temporal awareness provided by tickertape-enabled CVS having a differential effect related to experience, or simply that new developers are likely to increase the amount they write when making code changes anyway. It would be interesting to further explore this question in future work.

All but two of the developers either left no commit messages at all empty when checking-in code or decreased in the frequency of messages left empty. Of these two, developer 8 only had 6 null messages out of 1189 (0.5%); developer 2 was the lead developer on the project, who had at interview claimed that the 'empty message watcher script' had changed his behaviour, but was more likely to have changed the behaviour of others watching. This does seem to have been the case.

7. DISCUSSION

Communication and awareness have been identified as critical elements for successful software development. CVS has also been identified as an important coordination tool. Approaches to date for further supporting communication and awareness around the use of CVS have either involved the parallel use of mailing lists or the automated generation of tailorable visualisations of the code repository. The focus of this paper has been on the use of a lightweight tool that supports both the automated sending out of CVS messages to tickertape and the ability to conduct chat in the same place.

The volume of tickertape log data and the persistent level of voluntary use of it over an extended period of time demonstrates that the developers find it a helpful tool. This is supported by the comments reported in interview.

Analyses of the Orbit and Elvin CVS logs further demonstrate the importance of tickertape-enabled CVS in supporting awareness and communication, influencing developers to write more informative comments when checking-in code, and when augmented with the 'empty message watcher script', encouraging the 'good' software engineering practice of not leaving comments blank.

We had initially expected to find qualitative differences in the types of messages written after CVS was instrumented to send out tickertape notifications, reflecting the developers' view of CVS as more of a communicative tool. However, we found little evidence of such a change. We suggest that given the context in which CVS messages appear, in a shared interface with chat and other messages, it is unnecessary for developers to be explicitly communicative in the types of message they write. We are currently carrying out an in-depth analysis of logs of all messages sent to tickertape. Preliminary findings suggest that CVS messages are frequently embedded in ongoing conversations, where they are used as a shared informational and communicative resource by both the developer checking-in code and by others.

This leads us to why the combination of Elvin/Tickertape and CVS has been so successful in this organisation: it gives timely perceptual form to information that can be used as resource by the developer group that previously required explicit effort to find out. Furthermore, given that it relies on existing tools (CVS, Tickertape) and existing effort (checking code back in and entering comments), this information is essentially produced for free.

Crucially, CVS messages are provided in the same interface as chat messages, facilitating discussion about and negotiation around code changes.

Other features of the tickertape interface encourage its wide adoption across the organisation: it takes up minimal screen real estate so is more likely to be running and in view; it tends to sit at the periphery of attention by being position at the top or bottom of the screen; it supports a discretionary model of use – one can choose to attend to it or not and there are other mechanisms for catching up if something is missed (e.g. a threaded chat interface, the tickertape log, the CVS log); there is no requirement to clean up or manage messages once a profile has been set up as messages have a user-defined timeout for staying on tickertape; and subscriptions can be filtered as required on content to deal with potential overload issues.

It remains to be seen whether the lightweight, generic approach of tickertape-enabled CVS would extend beyond the research environment where it was developed. Indications are that it might: while the people who are the user group in this study are also the developers of the tool they are using, this instrumentation of CVS to make use of Elvin/tickertape is incidental to their main work, i.e., there was no organisational incentive or sanction for them to use the tool. To reinforce that this is not simply an effect of developers 'needing' to be seen to use their own tools, other software development groups that are not part of the same organisation have been using similar Elvin/tickertape augmentation of CVS for a number of years (although we do not have access to their log data). There is also the fact that this tool has remained in use over many years and with many different people coming and going in the team.

However, it does remain unclear whether it would scale to much larger commercial development projects, and what the trade-offs would be between the lightweight, generic, flexible approach and more heavyweight dedicated systems offering more functionality, such as the overview that visualisations offer, but with greater costs in terms of effort, screen estate, etc. We see this as an opportunity for future research.

8. CONCLUSION

In this paper we have described how a generic notification system called Elvin and a tickertape interface came to be used to support awareness of code changes in a CVS repository. Data from interviews and a high-level log analysis demonstrate that the tool was effective in supporting timely interaction around CVS entries.

Analysis of the CVS logs of two different projects show that when using this tool, developers tended to provide more information about code changes when checking-in code. We argue that this reflects changes in developers' attitudes to CVS logs as an informational resource.

This research shows that a lightweight tool that integrates notification and chat facilities with CVS can change the practices of the developers around use of CVS and enable new forms of conversations within the software development process.

Future work will focus on both the CVS and tickertape logs, to look at how the public availability of CVS together with the facility to chat affected working practices, for example management and articulation of work flow, task coordination, improved awareness, and social interaction. This paper has demonstrated how CVS messages became longer and more descriptive after the introduction of Elvin. Future work will explore whether this finding is reflected in the more general use of 'chat' around CVS commits, and what affects did tickertape as a communication tool have on the conversational practices within the software development team?

9. ACKNOWLEDGMENTS

We would like to thank Bill Segall, David Arnold, Ted Phelps, and other members of the Elvin development team for fruitful discussions, participation in interviews and access to the CVS and tickertape logs.

We would also like to thank John Halloran, Eva Hornecker, and Hilary Smith for their comments on a draft of this paper.

10. REFERENCES

[1] B. Berliner. CVS II: Parallelizing software development. In USENIX Association, editor, *Proceedings of the Winter 1990 USENIX Conference,* (Washington, DC, USA, January 22-26, 1990), Berkeley, CA, USA. USENIX. 341-352

[2] Chiang, R. and Mookerjee, V. S. Improving software team productivity. *Commun. ACM, 47, 5,* (May 2004), 89-93.

[3] Churchill, E. F., Trevor, J., Bly, S., Nelson, L., and Cubranic, D. 2000. Anchored conversations: chatting in the context of a document. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '00) (The Hague, The Netherlands, April 01 - 06, 2000). ACM Press, New York, NY, 454-461.

[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] Fitzpatrick, G., Parsowith, S., Segall, B., and Kaplan, S. 1998. Tickertape: awareness in a single line. In *Conference Summary on Human Factors in Computing Systems* (*CHI 98)* (Los Angeles, California, United States, April 18 - 23, 1998). ACM Press, New York, NY, 281-282.

[9] Froehlich, J. and Dourish, P. 2004. Unifying Artifacts and activities in a visual tool for distributed software development teams. In *Proceedings of the 26th International Conference on Software Engineering (ICSE 2004)* (Washington DC, USA). IEEE Computer Society, 387-396

[10] Grinter, R. E. Using a configuration management tool to coordinate software development. In *Proceedings of Conference on Organizational Computing Systems* (Milpitas, California, USA, 1995). ACM Press, New York, NY, 168-177

[11] Gutwin, C., Penner, R., and Schneider, K. 2004. Group awareness in distributed software development. In *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work* (CSCW'04)(Chicago, Illinois, USA, November 06 - 10, 2004). ACM Press, New York, NY, 72-81.

[12] Herbsleb, J. D. and Grinter, R. E. Splitting the organization and integrating the code: Conway's law revisited. In *Proceedings of the 21st international Conference on Software Engineering* (ICSE 1999) (Los Angeles, California, United States, May 16 - 22, 1999). IEEE Computer Society Press, Los Alamitos, CA, 85-95.

[13] Herbsleb, J.D. and Grinter, R.E. 1999. Architectures, coordination, and distance. *IEEE Softw. 16, 5,* 63-70

[14] Hupfer, S., Cheng, L., Ross, S., and Patterson, J. 2004. Introducing collaboration into an application development environment. In *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work* (CSCW '04) (Chicago, Illinois, USA, November 06 - 10, 2004). ACM Press, New York, NY, 21-24.

[15] Kraut, R. E., Fish, R., Root, R. W. and Chalfonte, B. Informal communication in organizations: form, function and technology. In Oskamp, S. and Spacaman, S. (Eds) *People's reactions to technologies in factories, offices and aerospace.* Sage Publications, 145-199

[16] Kraut, R. E. and Streeter, L. A. Coordination in software development. *Commun. ACM, 38, 3* (March 1995)*,* 69-81.

[17] Perry, D., Staudenmayer, N. and Votta, L. G. People, organizations, and process improvement. *IEEE Software, 11, 4* (July 1994) 36-45

[18] Sarma, A., Noroozi, Z., and van der Hoek, A. 2003. Palantír: raising awareness among configuration management workspaces. In *Proceedings of the 25th international Conference on Software Engineering* (ICSE 2003) (Portland, Oregon, May 03 - 10, 2003). IEEE Computer Society, Washington, DC, 444-454.

[19] Segall, B. and Arnold, D. "Elvin has left the building: A publish/subscribe notification service with quenching," *Proceedings AUUG Technical Conference (AU*UG'97) (Melbourne, Australia, September 1997) 243-255

[20] Segall, B., Arnold, D., Boot, J., Henderson, M., and Phelps, T. Content based routing with Elvin4. In *Proceedings AUUG2K* (Canberra, Australia, June 2000).

[21] Storey, M. D., Čubranić, D., and German, D. M. 2005. On the use of visualization to support awareness of human activities in software development: a survey and a framework. In *Proceedings of the 2005 ACM Symposium on Software Visualization* (SoftVis '05)(St. Louis, Missouri, May 14 - 15, 2005). ACM Press, New York, NY, 193-202.

[22] Strauss. A. The articulation of project work: An organizational process. *The Sociological Quarterly*, 29, 2 (1988) 163–178,.

[23] van der Hoek, A., Redmiles, D., Dourish, P., Sarma, A., Silva Filho, R., de Souza, C. Continuous Coordination: A New Paradigm for Collaborative Software Engineering Tools, Workshop on Directions in Software Engineering Environments (WoDiSEE 2004), held in conjunction with the 26th International Conference on Software Engineering (ICSE 2004—Edinburgh, Scotland), May 2004, pp. 29-36

[24] Yamauchi, Y. Yokozawa, M., Shinohara, T. and Ishida, T. Collaboration with lean media: how open-source software succeeds. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work* (CSCW'00) (Philadelphia, Pennsylvania, USA, December 2-6, 2000) ACM Press, New York, NY, 329-338