In Proceedings of ECSCW'99, Copenhagen, Sept 1999, Kluwer Academic Publishers, pp 431-451.

Augmenting the Workaday World with Elvin

Geraldine Fitzpatrick, Tim Mansfield, Simon Kaplan, David Arnold, Ted Phelps, Bill Segall
CRC for Distributed Systems Technology, The University of Queensland, Australia 4072. Ph: +61-7-33654310. Fax: +61-7-33654311.
Email: {g.fitzpatrick,t.mansfield,s.kaplan,arnold,phelps,bill}@dstc.edu.au.

This paper addresses the problem of providing effective, computer-based support for awareness and interaction in the distributed workaday world. We report the story of how our content-based pure notification service, called Elvin, became widely adopted in our organisation and elsewhere, augmenting the virtual work environment, and providing perceptual resources for awareness. Examples of its uses include: support for interaction via bi-directional chat-like facilities as well as support for uni-directional notifications, for example push-based information from services such as WWW and email, and notifications of the activities of others through rooms bookings, version control changes, and so on. These uses have had a significant impact on the way people interact with information sources and on social cohesion within the organisation. The attraction of Elvin lies in its conceptual simplicity, absence of built-in policy, expressive power and multilingual range of simple APIs. Its uptake is largely a result of the Tickertape Elvin client, which provides a simple, compelling interface usable in numerous different situations. We contend that even though it does not try to be a collaboration-friendly notification service, Elvin is paradoxically very useful for collaborative awareness and interaction support.

Introduction

Events, such as people arriving or leaving, phones ringing, sighs of frustration, the grind of the coffee machine, the hum of the printer, provide us with the per-

ceptual resources we need to maintain awareness of our environment and of the possibilities we have for interaction with others. We use relevant events to coordinate our work or play, arranging our activities to take account of the exigencies of the moment and work toward our goals, while at the same time filtering out irrelevant events. Frequently this contingent arrangement takes place with little or no conscious effort (Heath and Luff (1992) and Robertson (1997) give more detailed discussions of these phenomena).

In the virtual world, events that would keep us informed of what's going on 'around us' and of who we could interact with become imperceptible. We don't know a file has changed, a colleague has arrived at work or a new directory has been created, and so on, unless we explicitly think (or know) to check. Thus the low-effort mechanisms through which we coordinate our activities in the real world are unavailable to us. That is, there are no inherent mechanisms in the virtual realm by which these actions or events can be made available as a perceptual resource (Robertson 1997) for awareness.

With the growing appreciation for the importance of awareness in promoting a sense of shared place and shared work, there has been an increasing emphasis in the CSCW community on how to provide effective computer-based support for awareness in distributed environments. This has typically taken the form of event-based notification services (Ramduny et al. 1998). However, most tend to be designed to support synchronous collaborative applications, either as an application-specific service (Fuchs et al. 1995) or as a generic service to a single class of applications, for example (Hall et al. 1996; Patterson et al. 1996).

Rather than simply trying to build targeted collaborative environments, our focus is on how we might go about making the workaday world (Moran and Anderson 1990) more 'inherently' collaborative. When much of that workaday world happens within a distributed virtual and physical environment, as it does in our organisation, the DSTC[1], the question becomes one of how can we instrument and augment everyday working tools to give events a virtual presence and to support social interaction.

This paper concerns the design, uses and possibilities for presence and awareness support of a notification service called *Elvin* (Segall and Arnold 1997). Elvin is a generic service middleware designed for distributed systems. Rather than designing it to be domain-specific; Elvin's designers attempted to design a *general-purpose notification service*.

After its construction, it began to be spontaneously used for new purposes by different research and prototyping groups within our organisation, including our own Orbit project (Mansfield et al. 1997). The technical attraction of Elvin lies in its conceptual simplicity, its absence of built-in policy, its expressive power and its multilingual range of simple APIs making it usable by both users and pro-

[1] Distributed Systems Technology Centre

grammers of many different biases and skill levels. Its uptake is largely a result of the Tickertape Elvin client, which provided a simple, compelling interface usable in numerous different situations.

We contend that even though it does not try to be collaboration friendly, Elvin (and other pure notification services (Ramduny et al. 1998) like it) is paradoxically very useful for collaborative awareness and interaction support. We will show that this is because it allows us to *augment* the workaday world to provide the perceptual resources for awareness. Its design makes it easy to produce information about events in the virtual (and even, in some cases, the physical) and to select relevant information.

The discussion is structured as follows. First, we outline the design of Elvin, its features and conceptual model. Secondly, we summarise related work. Thirdly, we describe some user and programmer experiences with Elvin gathered by interview from around our organisation. Fourthly, we discuss how the design of Elvin facilitates those experiences and we argue why we think that pure notification services are particularly effective for CSCW. Finally, we discuss future work.

The Elvin Event Notification Service

The function of a notification service is to act as a distributor for descriptions of *events* or *notifications*. We define an event as any significant change in the state of an observed object. *Producers* detect events (and are responsible for determining that the status change is significant), and send descriptions to the notification service for dissemination to interested *consumers*, as shown in Figure 1.

Elvin is a 'pure' notification service (Ramduny et al. 1998): producers send notifications to the service, which in turn sends them to consumers. The notifications describe events using a set of named attributes of simple data types and consumers subscribe to a 'class' of events using a boolean subscription expression.

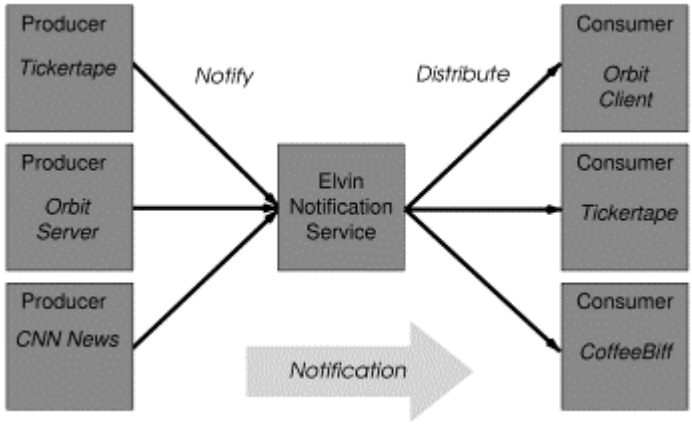

Figure 1. Elvin Architecture Overview.

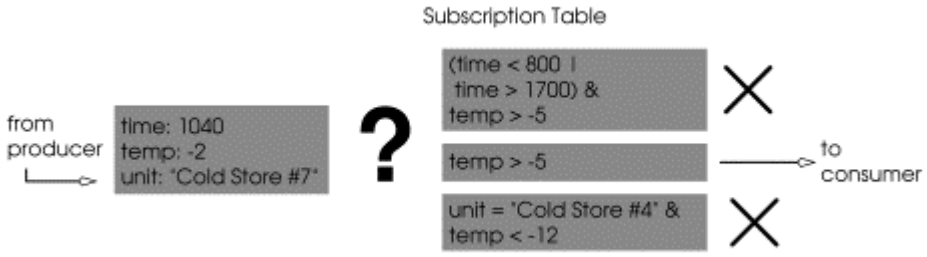

Figure 2. Subscription examples for an environmental monitoring system.

When a notification is received at the service from a producer, it is compared to the consumers' registered subscription expressions and forwarded to those whose expressions it satisfies. An example of this is given in Figure 2.

This *content-based* selection of notifications is often sacrificed by other notification services in favour of less flexible mechanisms because it is difficult to implement efficiently. A common alternative uses named 'channels' or 'topics' that must be specified by both the producer and consumers. A key benefit of content-based notification is the absence of this 'coupling' between producers and consumers, promoting system evolution and integration.

Once producers are freed of the responsibility to direct notifications, the determination of the significance of a state change becomes less important: they can promiscuously notify any potentially interesting information, and rely on the notification service to discard notifications of no (current) interest to consumers.

While large volumes of unused notifications may be useful from a user's perspective, they consume network bandwidth. To overcome this problem, Elvin includes a *quenching mechanism* which allows producers to discard unneeded notifications without sending them to the server (see Segall and Arnold 1997 for more detailed discussion).

In order to support organisation-wide notification, the implementation of the notification service must cater for many client applications. A single Elvin server can effectively service thousands of clients (producers or consumers) and evaluate hundreds of thousands of notifications per second on moderate hardware platforms. Further, additional servers can be configured in a *federation*, sharing the load of notification delivery, providing wide-area scalability (allowing Elvin to work across multiple LANs) and ensuring fault-tolerance in the face of individual server failures.

Like other pure notification services, Elvin does not store the notifications it sends – it provides no persistence for notifications.

The Elvin server is implemented in C for Unix platforms, and client-side libraries are available for C, TCL, Smalltalk, Python, Lisp and Java. The basic Elvin services are implemented via several simple library calls that support session setup and termination, producer notification, consumer subscription and callback registration, polling (where required) and quenching.

Producer and consumer commands make the service accessible from a command line shell and from shell scripts. Non-programming users can use graphical tools like Tickertape (see below) to produce and consume certain kinds of notifications.

Related Work

Numerous workplace studies (for example, Heath and Luff 1992) have identified awareness as being crucial for the dynamic coordination of ongoing work. The approaches to computer-based support for awareness are many and varied. Dourish and Bellotti (1992), for example, identify three mechanisms: explicit informational awareness mechanisms (such as those provided by version control system annotation); role-restrictive mechanisms (such as those used in some group editing systems and later in many workflow systems); and shared feedback mechanisms (where information about actions or events is collected and presented as background information in a shared workspace). More recent research has, as observed by Sandor et al (1997), evidenced an overwhelming concern with such event-based shared feedback approaches to awareness, which are more flexible than role-restrictive approaches.

Sandor et al provide a critique of event-based approaches which only permit subscription based on event types, arguing that the need to define event types *a priori* make such systems inflexible. By using content-based subscription, Elvin avoids this problem.

Previous work on notification services for awareness within the CSCW community has tended to focus on the support of synchronous collaborative work. The term is used to refer to a variety of systems with very different behaviors and which may provide notifications directly to users or only to applications or both.

Ramduny et al (1998) introduce a taxonomy for characterizing notification services based on their communication behavior and the level at which they operate (system or user). The authors distinguish between services primarily on how they enable a client that effects a change in some piece of data (the active client) to communicate information about that change (the notification) to a client that wants to know (the passive client). They also discuss the possibilities of differences in pace and volume between system level services and corresponding user level services.

In the terms offered by Ramduny et al, Elvin is a pure notification service (active client tells notification service, notification service tells passive client) which remains completely separate from the observed data. Elvin primarily serves the system level, offering support for applications to exchange notifications but little explicit support for user notification and volume or impedance matching.

Perhaps the best known CSCW notification service is Lotus PlaceHolder, which is based on Notification Service Transfer Protocol or NSTP (Patterson et

al. 1996). Rather than providing a pure notification service, the designers of NSTP opted to focus on facilities for synchronous collaborative applications. They therefore conflate a notification service with a centralized data store for shared data. The work primarily focuses on providing a protocol (based loosely on HTTP) for interoperation between notification services. PlaceHolder is a sample implementation of such a service. The notification service includes a number of design notions such as, Things (roughly, application objects) in Places (generalization of application session), with Facades (which moderate access to Things). The service is also envisioned as providing some system and some user level services ("place browsing" allows users to move between Places).

The design of NSTP means that PlaceHolder will primarily be useful for the construction of bespoke synchronous collaborative applications. The explicit centralization of shared data makes it difficult to integrate PlaceHolder with existing applications. The complexity of the required implementation also makes it difficult to produce competing servers for PlaceHolder to interoperate with. In contrast with Elvin, the service is also essentially channel-based and no concessions to scalability appear to have been made.

Hall et al (1996) also focus on the design of a shared data communication service for synchronous groupware with CORONA. The designers of CORONA however, partition the system into a number of services and maintain the 'publish-subscribe service' as a pure notification service using a channel-based approach. Optimized for wide-area use, the publish-subscribe service multicasts published notifications to distributor nodes which in turn multicast to other distributors which then send on to local subscribers. This enhances scalability by "minimizing system-wide awareness and change". This use of multicasting is not available to systems such as Elvin which do not rely on channel-based subscription.

Lövstrand (1991) describes the Khronika system, which for several years was in use at Rank Xerox EuroPARC (Now Xerox Research Centre Europe, Cambridge). Khronika was fundamentally an event database that stored user-level events (meetings, brown-bag lunches, etc.). Users could discover events by browsing the database or by assigning 'event daemons' to issue notifications when certain kinds of events triggered. The system is very relevant to our discussion because it had a constraint language allowing users to specify quite complex subscriptions to events in which they were interested. We can consider Khronika's constraint language analogous to Elvin's subscription language.

Khronika is also interesting because it was primarily a user-level notification service not a system-level service. Despite its intended domain, its design was similar to Elvin (which was developed much later by designers with no knowledge of Khronika).

Using Elvin for the Workaday World

As indicated above, Elvin quickly began to be used by a number of different research and prototyping groups within our organisation. A key trigger in facilitating this usage was an incidental application called Tickertape (Fitzpatrick et al. 1998; Parsowith et al. 1998) (to be discussed below) that developers initially created to give a visual display of event traffic. Tickertape enabled people to quickly understand what Elvin was, how it worked, and how it could be used for their own purposes.

The usage examples that we discuss here were thought of and/or developed by a wide variety of people most of whom were not directly involved in the Elvin project, most of whom were technical developers and some users. Some of the systems, applications or tools take advantage of the Tickertape interface. Others use Elvin directly, plugging it into some other piece of software. Some are primarily to support 'individual' tasks. Others are to support collaborative aspects of work. Some started out to support an individual and were soon found to be useful as a way of providing others with the information they needed to better coordinate their work. The authors are also active users or developers of Elvin and/or its many associated tools.

In the following section, we briefly discuss the methodology for this study. We then separate the following discussion into *bi-directional* uses (where users themselves can generate messages as notifications as well as receive notifications, supporting awareness through interaction) and *uni-directional* uses (where notifications are generated from some external source, supporting awareness via information flow) (Parsowith et al. 1998).

Methodology

The following usage comments are an aggregate of a Tickertape usage study reported in (Parsowith et al. 1998) (relying on semi-structured interviews and some conversational analysis based on 20,000 element log-files) and a further survey and semi-structured interviews conducted at the time of writing. In our initial study, Tickertape had roughly 20 users, usage has risen slowly in the last year to roughly 40 users.

The analysis of that data has been informal, without reference to a theoretical framework.

Bi-Directional Uses of Elvin

Three key tools give users access to Elvin for bi-directional use: Tickertape, Tickerchat and CoffeeBiff.

Figure 3 The Tickertape interface showing exemplary scrolling messages

Tickertape

Tickertape is a highly tailorable tool that uses Elvin. It is both a producer and a consumer of notifications. It displays notifications that the user subscribes to and it can be used to construct chat-style messages that are sent as notifications.

The Tickertape interface, as shown in Figure 3, consists of a single resizable, rectangular window, showing small, colour-coded messages that scroll from right to left. Each message corresponds to an Elvin notification of a specific format that has been received by the Tickertape application. For example, the left-most message in the figure is from user 'arnold', has been sent to the group 'b&d', and has the text 'so my monitor works'. The Tickertape is designed to take up minimal space – the active area is a single line, and borders, etc., can be removed to make the Tickertape 'fade into the background'. Most users position their Tickertape(s)[2] on the edges of their screens, where they provide a simple kind of peripheral access to the information that scrolls by.

Tickertape users subscribe to messages at two levels: they indicate the 'groups' they are interested in, where group is an attribute contained in the content of all messages to which Tickertape can subscribe, and they indicate some filters over the contents of messages which have the appropriate group attribute values. A simple set of dialog and menu list boxes is used to allow non-programmers to perform this customisation. If events have a MIME attachment, associated graphics on the scrolling marquee indicate this, and the user can trigger the attachment with a mouse click.

Individual notifications have a lifetime over which their appearance fades from colour to grey, thus providing an indication of how timely the information is. The lifetime is user-defined for each group. Users can also choose to delete or save a scrolling message by clicking on the message itself. Tickertape therefore provides users with a mechanism for controlling the transience of information.

Tickertape provides no persistent storage of notifications, once a message fades away – it is lost.

One of the most popular uses for Tickertape is as a lightweight channel-based *semi-synchronous chat tool*. Users can define new chat groups by agreement with their peers. Messages can be sent as Elvin notifications by clicking on the Tickertape and using the resulting pop-up dialogue shown in Figure 4. Messages are

[2] One of the authors of this paper runs three tickers at all times, each customised to different types of information and running at different speeds.

received as text on the Tickertape (Figure 3 shows an example conversation).

Examples of interactive groups are: the 'Chat' group for all the people in the organisation (used as a general discussion forum and for user-generated announcements), the 'lunch' group (used by regular lunch-goers to organise lunch times and venues), the 'b&d' group (for Bill and David who are working closely together on a project) and the Elvin group (for the developers of Elvin).

Such chat groups are used extensively within the organisation by both technical and non-technical staff, especially as people are distributed across different offices on different floors and, for a time, in different buildings. Tickertape is also used by people working from home to interact with colleagues in the office. In fact, the authors of this paper used it to discuss the paper while one was located in another country. It supports work-related discussions – a recent newcomer used the Chat group to ask for help in setting up his new environment and finding out what services were available. It is also used for social chitchat, and for broadcast announcements, such as "paging Dr Tim" (acting as a silent public announcement or paging system).

Interactions over these bi-directional groups tend to be spontaneous, short, informal, often irreverent, and 'bursty', similar in nature to face-to-face conversation. They incorporate the synchronicity and immediacy of the telephone with the asynchronous style of email and are especially suited for temporally relevant information, such as "there are cakes in the kitchen".

Very quickly, Tickertape has become embedded into the normal working environment of the organisation as yet another means for communication and interaction along with the telephone, email, face-to-face discussions, and porthole video images. Each of these has particular uses for which they excel and Tickertape has found its niche amongst them. It is not uncommon to see a discussion over Tickertape that ends with a comment such as "uh oh, see email…" or "wait a minute … I'm coming down" when the content of the discussion becomes more detailed than can be usefully handled in short chat messages.

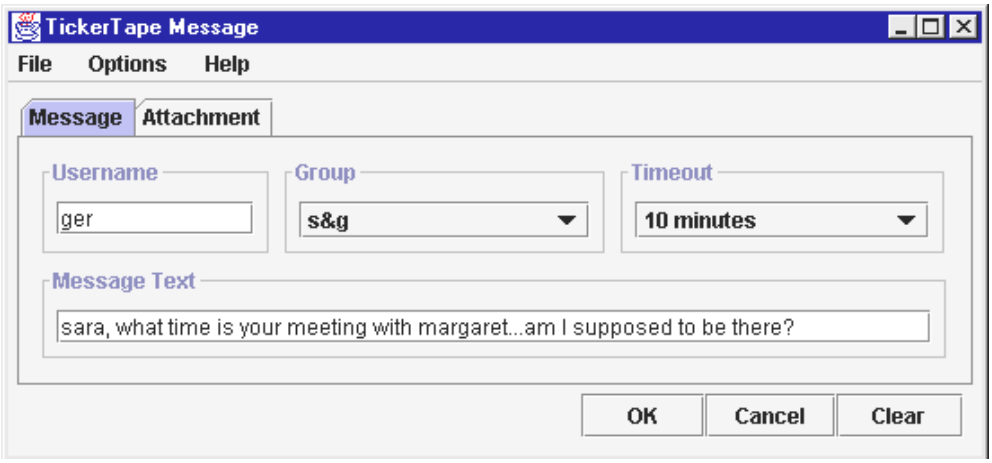

Figure 4 Sending a chat message via the Tickertape Dialog

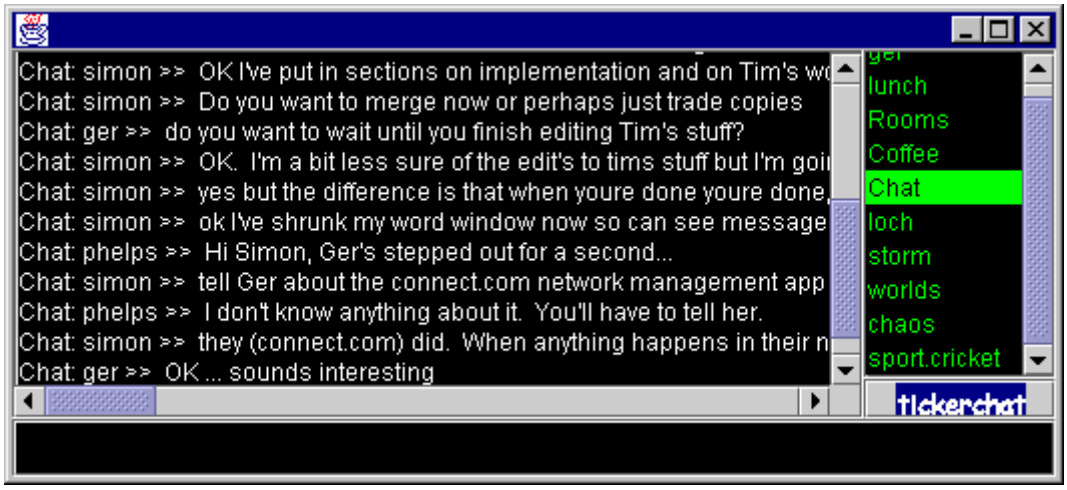

Figure 5 A Tickerchat discussion. One person was at home, the others are in the office.

Tickerchat

For various reasons, three separate programmers developed a chat-tool-style interface to Tickertape. All of these *Tickerchat* implementations provide the same basic interface, shown in Figure 5: a large window in which previous messages are shown, a line at the bottom in which one may enter new messages, and some way of selecting the group to which one's messages should be directed. Messages shown in Tickerchat do not fade, they simply scroll up the transcript window. Users can choose to clear the buffer or save the buffer when they no longer want to see the message stream.

The Tickerchat tool extends the transience of the information in Tickertape making it more convenient for discussion as opposed to notification. This persistent quality allows users greater discretion about when they attend to discussions because they know they won't miss anything through timeout. One developer was motivated to build a Tickerchat by his frustration at regularly finding, upon returning to his desk after an absence, a raging debate on the Tickertape in which the original issues had long since faded out.

Figure 6. CoffeeBiff icon with number and scrolling names.

Many users now choose to run both a Tickertape and a Tickerchat, using the Tickerchat when they enter a discussion and reserving the tape for observation of the flow of information.

Churchill and Bly (1999) study the use of a MUD for informal communication and coordination in a laboratory quite similar to our own. While Tickertape and Tickerchat provide affordances similar to those of the MUD, with similar positive experiences to those reported by Churchill and Bly, we have avoided some of the pitfalls that they describe, notably the problems with participating in multiple ongo-

ing chats simultaneously, and the disconnection between the MUD and the user's real-world tools. Our tools avoid problems with multiple channels by simply providing access to multiple channels of conversation on different ticker channels. By designing Tickertape in particular as a peripheral application that typically sits on the top or bottom edge of the screen, we obviate the need to always switch context between the chat tool and other tools

CoffeeBiff

A different example of a bi-directional use of Elvin is *CoffeeBiff*. When people go to the kitchen for a coffee break, they can click on the CoffeeBiff icon to indicate their intention to colleagues, as shown in Figure 6. If a person knows when her friends from other parts of the building are taking a break it can help her to coordinate her break times, or to identify when a colleague will be in the kitchen and possibly free for a casual chat. A background application subscribes to CoffeeBiff notifications and sends a *second-level* notification to the Tickertape "Coffee" group when more than five people are drinking coffee, indicating that some kind of party is clearly in progress.

Uni-Directional Uses of Elvin

Another significant use of Tickertape has been for uni-directional events where pre-existing external event streams are instrumented to produce Tickertape notifications. This supports awareness through notification, providing content-based information filtering. Event streams in regular use include: postings to Usenet news groups, commercial news sources on the World Wide Web (WWW), personal email, and other sources such as a schedule system. Various developers around the organisation have written programs to translate an event source into an Elvin notification.

Content Filtering

Users can subscribe to these event sources by content. This means, for example, that a user can essentially say "show me all the postings to comp.groupware that mention 'shared drawing tool'" or "show me all the ABC headline news postings except for sport". Whenever a corresponding posting is detected, a Tickertape message will display the subject header with the text of the new article attached in a MIME attachment. The attached article can be viewed with a mouse click.

Users have also used the content-based subscription feature to implement a form of social filtering of information. For example, several people have subscriptions which mean "if this person from our organisation posts to the (newsgroup), tell me about it". This is making use of what they know about others' shared interests and areas of expertise – it is highly likely that if that person is contributing to a thread, then it will be of interest to them as well.

Users who take advantage of Usenet, WWW and email filtering in this way uniformly report that the way in which they interact with external information sources has been radically altered. Many users stated in interviews that the only time they go to a Usenet newsgroup now is when a Tickertape notification indicates there might be a thread of interest to them. Others never go to the source but read postings via Tickertape only.

Email Notification

There are now several ways for users to be alerted via Tickertape that they have received e-mail. Users can add a script to their mail processing (`.forward`) file which is run whenever they receive an email. This script generates a notification with appropriate summary information (sender, subject, and MIME information to access the message), and sends it to a private Tickertape group. More sophisticated versions of the script can be used in conjunction with automated mail filtering and sorting programs such as PROCMAIL. A user can access the email message directly by using the MIME attachment to open the appropriate folder and message. By enriching the messages generated by the mail script so that it contains more of, or all of, the mail message, users can then use the content filtering capabilities of the subscription language to filter the mail-related notifications they see on the Tickertape.

Many users report that email is now far less of a distraction since they can see immediately who an email is from, and what its subject is, and thus make a quick judgment about whether it is important enough to stop what they are doing to go read it.

Advance Schedule Warning

We have already noted that Tickertape is particularly useful for presenting temporally relevant information of transient importance. This feature has been exploited in the instrumentation of the DSTC Rooms Booking calendar. At ten minutes prior to the booked time, a Tickertape notification is generated stating the time, room, and meeting description. Most obviously, this serves as a useful reminder to people. More importantly though, it serves as a general mechanism for making activities within the organisation more visible, and giving people a way of ‘keeping an eye’ on activities that they might not otherwise have known about.

Making Actions Visible

There are many examples of users writing code to generate events that reflect their activities in the virtual computing environment. One user, for example, has written scripts to generate a notification whenever he logs into or out of his workstation. In this way, others know when he has arrived at or leaves from work.

There are also ‘file-watcher’ and ‘web-watcher’ event generators that monitor changes to the network file stores and local web pages respectively and send out

notifications via the relevant Tickertape groups. These serve as an important information source for people who rely on system files, maintain sections of the company Web, etc.

Analogous information is available via notifications generated from CVS. CVS is a revision control package used extensively within the DSTC to manage source code. Typically, a programmer will have a copy of the source tree of a project 'checked out'. She keeps the copy synchronised with the copies of other programmers by occasionally updating her copy in order to incorporate their changes and commit her own changes. As part of the 'commit' process, the user is prompted to log a comment as to what has changed and why. These comments play an important role in the coordination and articulation of software development by teams because others can go to the log file and read about what changes have happened (see Grinter 1997).

To enhance out-of-band communication among developers, and facilitate early identification of conflicts, we have extended CVS to generate a Tickertape notification that includes the commit comments whenever a commit is performed. In addition to providing warnings about various kinds of commit problems, such as a race condition, this informs the other programmers who subscribe to the 'CVS' group of what changes other users are making. In effect this makes visible the previously invisible activities of the user's colleagues. This allows the group of developers to contingently rearrange their work, negotiate to resolve problems, offer warnings and suggestions, and so on, all in a more timely manner than previously possible.

Other Elvin Applications

Other applications have been developed using Elvin as the underlying event notification structure. We will mention two here:

ORBIT: Although we described Elvin as generic notifications service rather than a service for a bespoke collaborative system, it can still be used in collaboration-specific environments as we have done with Orbit (Mansfield et al. 1997).

Orbit is a collaborative desktop environment that has a client-server architecture. When a client initiates interaction with the server, it uses a remote procedure call (RPC) mechanism (CORBA) to send a request to, and receive the reply, from the server. If the server initiates communication, it is because the server's state has changed and clients might need to be notified. In this case, it sends a notification of the state change via Elvin. This offloads the responsibility from the server to Elvin for keeping track of which client is interested in what information.

Tickertape has also been extended so that bi-directional groups can be created that correspond to a group zone, thus providing Orbit users with a low-bandwidth communications tool that is specific to their work context. By adding additional information to the notifications that the Orbit server sends to its clients, we were

able to display these notifications in Tickertape as well. For example, when the membership of a group zone changes, the same notification that is sent to update the state of the client can also be displayed on the Tickertapes of the members of that zone to tell them of the new (or lost) member.

Network Management: An Internet service provider runs status on each of its servers which then forward information via Elvin to diagnostic correlation engines. When problems with the status of a server are identified (or diagnostic messages fail to arrive), events are sent via an Elvin-to-pager gateway to the pagers worn by the sysadmins on duty.

Summary

In this section, we have shown a variety of ways in which Elvin has been used within the organisation. The chat-based tools and Coffee Biff tool that use Elvin for both production and consumption of events (we called this 'bi-directional') have been significant additions to existing communication resources for supporting interactions, both for work and fun, in a distributed workplace. Not only have information and expertise been shared, but social cohesion and relationships have also been strengthened.

The uni-directional uses where Elvin, mostly via Tickertape, has been used to push information to users from externally produced event streams has radically changed the ways in which people go about accessing and using many of these external sources (such as email, WWW, Usenet news etc.). Other uni-directional uses, for example, where CVS or the Room Booking systems have been instrumented, have facilitated opportunities for far greater awareness of activities and events both within the virtual realm and the social/organisational realm than has otherwise been possible. This is because Elvin and Tickertape give a way to make the information available or 'present' as a perceptual resource for awareness.

Why Does Elvin Work?

Many of the cases given above are not intentionally CSCW-related, and this is precisely what makes them interesting to us. They are all examples of ever more powerful facilities being evolved through the incremental instrumentation or augmentation of workaday tools.

Moreover, the user community were prepared to put in this development effort because they believed, and indeed found, that the facilities would be useful for their for their everyday work (thus avoiding Grudin's (1994) work-benefit disparity problem). Perhaps one of the most surprising facts about the uptake of Elvin-based services around DSTC (and beyond) is that there has hardly been any effort to promote the use of the system; its uptake has been entirely spontaneous, as our users came to recognise and take advantages of the benefits.

The essence of Elvin's success is that it fulfilled a key need: providing a way to gather and redistribute collaboration-focussed information produced during the everyday work of our users. Another key element of its success was that it did not detract from or inhibit the use of existing tool sets, yet provided the ability to flexibly, spontaneously and incrementally construct a suitable workaday environment that afforded substantially improved awareness and interaction.

Gutwin and Greenberg (1996) define a taxonomy of information that users need to be aware of in a synchronous collaborative application. When the target domain is the larger context of all of the user's computer-based work, gathering this information is even more challenging. We cannot provide a collaboration toolkit and request that programmers use it, since they are not, in general, constructing applications which they would think of as collaborative. Indeed we cannot provide a collaboration environment and request that users use it either; users are going to use the tools that they need to accomplish their work, and shy away from tools that inhibit this accomplishment.

Using Gutwin and Greenberg's taxonomy, Elvin has been used to inform about participant, location, activity level, actions, changes and objects. In so doing, Elvin helped to make perceptible a significant portion of the 'event-based' cues which we use as perceptual resources for awareness to shape our workaday lives, and which have been missing in virtual (ie., computer-based) activities. It also facilitated lightweight informal interactions in a distributed environment.

Elvin Design Characteristics

We believe there are four *design characteristics* of Elvin that have contributed to its widespread use, specifically: simplicity; genericity; performance; and informality. These characteristics make the use of Elvin popular with authors of new software and straightforward to add to existing software. Tools constructed with Elvin also tend to exhibit these characteristics, which makes them popular with the broader user community. We will discuss these characteristics in turn.

Simplicity

The key design feature of Elvin is the simplicity of its API. Because the Elvin API is available for several programming languages, authors can usually use Elvin from their choice of development language. Because the API is simple (usually involving in the order of five method or procedure calls) and at a high-level of abstraction, it is an appealing choice for busy developers. Developers often try to use Elvin for tasks outside of its design criteria (such as mimicking RPC behaviour) for this reason alone. The designers often find themselves in the enviable position of explaining to hopeful Elvin users why they should **not** use it for their proposed task!

The service is also accessible from the command-line making it feasible to use

the service without extensive programming. Technically-aware users can easily write shell scripts to issue or subscribe to notifications. The log in/log out notification mentioned previously is implemented using shell scripts, for example. For the even less technically aware, GUI applications such as Tickertape provide an indirect way to use Elvin.

A secondary and related reason is that Elvin programming is fun. The various Tickertape and Tickerchats, CVS, mail handlers, etc., are all examples of developers building simple applications for sheer enjoyment, and these then propagating out into the wider community. This sense of fun and adventure soon passed from the narrower developer community to the larger community, including computer-phobic staff members (among them authors of this paper).

Genericity

Ramduny et al (1998) characterise notification services like Elvin (in which an active client tells the service about a change in some data and then the service tells a passive client about the change) as a "pure" notification service. Elvin lacks any inbuilt policies about information distribution and it stores none of the information it transmits. Its only purpose is the notification of events.

This means that Elvin can be used in any application that requires distributed notification, not just synchronous collaborative applications. It also means that it is not typically onerous to add Elvin code to (or 'elvinize') existing applications. Its lack of policy means that it is more likely to 'fit' into the target application.

The process of elvinizing an application is simply that of adding notifications wherever significant status changes occur in the application. Status change notifications allow any interested parties elsewhere in the network stay informed about the state of the application. This can be used to provide feedthrough (in Ramduny's terms) to other users or to share status with other applications or both.

Because Elvin is generic, programmers often add notification code to an application for reasons other than providing awareness. Some applications, for example Orbit, use Elvin as a means of implementing change notifications for a distributed Model-View-Controller architecture. Those notifications can still be used to gather awareness information even though it was not the programmer's.

Performance

The design of Elvin emphasizes efficient performance. The quench feature is designed specifically to minimize unnecessary network traffic. The consequence of quench (particularly when quenching is performed automatically by the producer API) is that developers can add notifications wherever status changes occur without regard to who might receive them now or in the future.

Quenching suggests potential applications such as using Elvin to generate debugging trace from immature applications. Rather than requiring the author to explicitly place conditional statements around trace generating code, they could

simply use Elvin notification code which would produce no trace unless some consumer (a transcript window perhaps) subscribed to it.

Quenching allows an application author to know that unnecessary notifications will not be sent. Combining quenching with other design characteristics such as genericity and simplicity means that it costs little more to include notifications of significant status changes. It is possible that this low cost may encourage programmers to include such notifications just in case the notifications become useful later.

Informality

Elvin notifications are not explicitly typed. Every notification is simply a tuple of attribute-value pairs. Elvin does not require any semantics (other than base type) be formally associated with any of these attributes or the tuple as a whole. Consumers subscribe to notifications based on content and typically select only what information they need from any notification they receive.

The power of such content-based addressing cannot be overstated. Application authors can include any significant information in a notification and add more information to the tuple as the application evolves. For example, the newswatcher was already generating events related to Usenet newsgroups. Once Tickertape came along, it was a simple matter to make it 'listen' to Usenet news events, and soon this became a favoured way of keeping an eye on a whole host of news services. Later, the newswatcher was modified to add mime-typed URLs to events to allow users to go directly to a news article from Tickertape.

Hence, a consequence of this design choice is that piecemeal growth and evolution of an event infrastructure across an organisation becomes feasible. All potential consumers will subscribe to the notification in the same way: using the Elvin subscription language. No *a priori* typing system – of message contents or structure as well as target addresses – needs to be imposed on the set of notifications an application can send. The informality also means that awareness clients can gather awareness information based on any information in the notification, not just based on what types of events the original authors thought might be interesting or useful. In this way awareness clients based on Elvin avoid problems associated with having to pre-specify all message types (Sandor et al. 1997).

The Tickertape GUI

Without the Tickertape interface, however, it is doubtful whether these design features alone would have made Elvin so popular among the user community. A significant effort in making any notification service more accessible for user-level awareness is the effort involved in building an appropriate GUI. With Tickertape, that effort has already been expended.

Despite some obvious shortcomings of Tickertape both as an application and

an interface, its main feature is that it exists. This makes Elvin accessible, understandable and usable by the broader user community. Once the user invests the initial effort to install and configure Tickertape, the subsequent barrier to usage for a whole variety of purposes (possible because of its tailorability) is very low.

It also promotes less than optimal strategies. For example, in our translation of commercial WWW news pages into Tickertape notifications, in many cases more specialised interfaces would probably be more suitable. In the room booking notifications example, it would be useful to have an interface that would allow one to cancel a booking. Our online notifications from the weather station could probably be better represented graphically than in a scrolling text line. On the other hand because of the difficulty of producing special-purpose graphical interfaces, it is unlikely that any of these interfaces would ever have been built. Because Tickertape was there, it was easy to add additional information sources and the interface via Tickertape was good enough, so the solutions worked well enough.

Summary

Elvin provides a simple way for application authors to provide information about significant status changes to other parties in the network. The potential ubiquity of the information and its lack of explicit semantics mean that gathering information about user presence, location, activity level, actions, the changes they are making and the objects they are acting on is feasible even from applications that are not designed for collaboration. This point is extremely important.

Conclusions & Future Work

The thesis of this paper has been that a generic notification service can be used to both enable users to maintain awareness of the activities or status of others and support interaction between users. It does this by providing users with a stream of information about events in the virtual and physical world and providing powerful tools for selecting relevant information. We've illustrated how a number of different event sources (for example, Web news, CVS, email delivery, information stores) can be instrumented to produce meaningful events, and how clients can be used to feed this information to users in a relatively unobtrusive fashion. Gluing this all together through content-based message subscription provides a high degree of flexibility, since new consumers or producers can be added independently of one another. Potentially any state-based information source or event stream could be instrumented in this fashion, and many different kinds of clients could consume these events. The limiting factor, as always, is whether the effort one must expend to get the notifications generated in the first place is worth the resulting benefits.

In our case we focused on information that was clearly 'out there' and reduced

the effort required by users to access it (by pushing it through notifications). A number of immediate benefits resulted. These include timely access to information and an ongoing perception of what individuals around the organisation are doing. The examples given clearly demonstrate that Elvin helps to do this, precisely because it augments people's real, workaday environment and allows them to share (or make visible) the resulting information with their peers.

Future work on Elvin will focus on infrastructure issues concerned with scaling and quench propagation, secure notifications and event correlation – the generation of higher-order events from patterns of simpler events. We also plan to continue to augment the working environment, on one hand through further instrumentation of applications and utilities. Much of this work we expect will continue the tradition of work of this kind in projects such as Khronika (Lövstrand 1991). We also plan to begin instrumenting the physical world, through the development of sensors and actuators that can transmit and respond to Elvin events. We hope this approach can begin to more effectively integrate the virtual and physical worlds.

Acknowledgments

Bill Segall and David Arnold are the key designers and developers of Elvin. Julian Boot has also contributed to the design and development of Elvin. We acknowledge Sara Parsowith for her work on earlier evaluations of Tickertape usage. We also thank the user community at the DSTC who contributed to the evolution of Elvin applications and appropriated it into their daily lives. More information about Elvin can be found http://www.dstc.edu.au/Elvin.

The work reported in this paper has been funded in part by the Co-operative Research Centre Program through the Department of Industry, Science & Resources of the Commonwealth Government of Australia.

This work has also been supported in part by the United States Defence Advanced Research Projects Agency under grants F30602-96-2-0264 and F30603-94-C-0161 (both administered by the US Air Force through Rome Laboratories). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, expressed or implied, of the Defence Advanced Projects Research Agency or the U.S. Government.

References

Benford, S., and Fahlen, L. (1993). "A Spatial Model of Interaction in Large Virtual Environments." *Third European Conference on Computer-Supported Cooperative Work (ECSCW '93)*, Milan, Italy, 109-124.

Churchill, E., and Bly, S. (1999). "Virtual Environments at Work: ongoing use of MUDs in the Workplace." *International Joint Conference on Work Activities Coordination and Collaboration*, San Francisco.

Dourish, P., and Bellotti, V. (1992). "Awareness and Coordination in Shared Workspaces." *CSCW'92*, 107-114.

Fitzpatrick, G. (1998). "The Locales Framework: Understanding and Designing for Cooperative Work," , The University of Queensland, Brisbane.

Fitzpatrick, G., Parsowith, S., Segall, B., and Kaplan, S. (1998). "Tickertape: Awareness in a single line." *CHI'98 Summary*, Los Angeles, CA, 281-282.

Fuchs, L., Pankoke-Babatz, U., and Prinz, W. (1995). "Supporting Cooperative Awareness with Local Even Mechanisms: the GroupDesk System." *Fourth European Conference on Computer-Supported Cooperative Work*, Stockholm, Sweden, 247-262.

Grinter, R. E. (1997). "Doing Software Development: Occasions for Automation and Formalisation." *Fifth European Conference on Computer Supported Cooperative Work (ECSCW97)*, Lancaster, UK, 173-188.

Grudin, J. (1994). "Groupware and Social Dynamics: Eight Challenges for Developers." *Communications of the ACM*, 37, 93-105.

Gutwin, C., and Greenberg, S. (1996). "Workspace Awareness for Groupware." *Common ground: CHI'96 Conference Companion*, Vancouver, Canada, 208-209.

Hall, R. W., Mathur, A., Jahanian, F., Prakash, A., and Rassmussen, C. (1996). "Corona: A Communication Service for Scalable, Reliable Group Collaboration Systems." *CSCW'96*, Boston, MA, 140-149.

Heath, C., and Luff, P. (1992). "Collaboration and control; crisis management and multimedia technology in London Underground line control rooms." *Computer Supported Cooperative Work*, 1(1-2), 69-94.

Lovstrand, L. (1991). "Being Selectively Aware with the Khronika System." *ECSCW'91*, Amsterdam, 265-277.

Mansfield, T., Kaplan, S., Fitzpatrick, G., Phelps, T., Fitzpatrick, M., and Taylor, R. (1997). "Evolving Orbit: A Progress Report on Building Locales." *Conference on Supporting Group Work (Group '97)*, Phoenix, Arizona, 241-250.

Moran, T. P., and Anderson, R. J. (1990). "The Workaday World as a Paradigm for CSCW Design." *CSCW'90*, 381-393.

Parsowith, S., Fitzpatrick, G., Kaplan, S., Segall, B., and Boot, J. (1998). "Tickertape; Notification and Communication in a Single Line." *APCHI'98*, Japan, 139-144.

Patterson, J. F., Day, M., and Kucan, J. (1996). "Notification Servers for Synchronous Groupware." *CSCW'96*, Boston, MA, 122-129.

Ramduny, D., Dix, A., and Rodden, T. (1998). "Exploring the Design Space for Notification Servers." *CSCW'98*, Seattle, WA, 227-235.

Robertson, T. (1997). "Cooperative Work and Lived Cognition: A Taxonomy of Embodied Actions." *Fifth European Conference on Computer Supported Cooperative Work (ECSCW97)*, Lancaster, UK, 205-220.

Sandor, O., Bogdan, C., and Bowers, J. (1997). "Aether: An Awareness Engine for CSCW." *Fifth European Conference on Computer Supported Cooperative Work (ECSCW97)*, Lancaster, UK, 221-236.

Segall, B., and Arnold, D. (1997). "Elvin has left the building: A publish/subscribe notification service with quenching." *Queensland AUUG Summer Technical Conference*, Brisbane, Australia.