Published in Proceedings of the ACM Symposium on User Interface Software and Technology, 1995, forthcoming.

Social Activity Indicators: Interface Components for CSCW Systems

Mark S. Ackerman
Brian Starr
Department of Information and Computer Science
University of California, Irvine
Irvine, CA 92717
ackerman@ics.uci.edu
http://www.ics.uci.edu/CORPS/ackerman.html
bstarr@ics.uci.edu

ABSTRACT
Knowing what social activity is occurring within and through a Computer-Supported Cooperative Work (CSCW) system is often very useful. This is especially true for computer-mediated communication systems such as chat and other synchronous applications. People will attend to these systems more closely when they know that there is interesting activity on them.

Interface mechanisms for indicating social activity, however, are often ad-hoc, if present at all. This paper argues for the importance of displaying social activity as well as proposes a generalized mechanism for doing so. This social activity indication mechanism is built upon a new CSCW toolkit, the Cafe ConstructionKit, and the Cafe ConstructionKit provides a number of important facilities for making construction of these indicators easy and straight-forward. Accordingly, this paper presents both the Cafe ConstructionKit as a CSCW toolkit as well as a mechanism for creating activity indicators.

KEYWORDS: computer-supported cooperative work, user interfaces, social activity, awareness, visualization, human-computer interfaces, information systems, CSCW

INTRODUCTION

Often you may want to know what the people around you are doing. For example, when working on a large team, it is energizing to know that people are busy working away. Or, one might like to know when his friends are available on a MUD, IRC, or other synchronous "chat" system. The exact semantics of their messages or documents may be less important than just knowing that some activity is occurring.

In general, CSCW systems need to convey an indication of their social world. User interfaces for groupware or CSCW applications must therefore convey this social information, which may vary according to the needs of the system. While many CSCW systems are partially aware of this requirement, particularly the need for shared representations, the need to reflect general activity is less understood. Additionally, implementation of this social indication is ad hoc when present. We would like to make such social indicators be a standard part of a CSCW application's user interface.

In the following sections, we argue that social indicators are useful in CSCW applications. We base this argument on a set of theoretical concerns from social psychology. Based on these theoretical concerns, we argue for the utility of a particular class of social indicator, which we call *social activity indicators*. They constitute a simple, yet powerful, mechanism for improving CSCW functionality.

We also show how easy it is to construct these social activity indicators, given the necessary tools in the underlying system. The generalized mechanism for indicating social activity uses a new CSCW toolkit, the Cafe ConstructionKit, that provides the building blocks necessary to easily implement social activity indicators. Accordingly, we discuss the Cafe ConstructionKit and then the indication mechanism built upon it. We conclude the paper with some examples of social activity indicators.

PROBLEM

Imagine yourself a member of a software engineering team working with both individual tools as well as group tools. You are, as usual, under extreme time pressure to make your deadlines. Perhaps you are even working on multiple projects with multiple teams. Consider the following seemingly unrelated problems:

- To which applications do you give valuable screen real estate?
- How do you know when it's time to really work and when it's time to ignore requests to work on a specific project?
- Where do you ask questions and of whom?
- How do you know when there will be enough users on a group system to bother using it?

There is a well-known problem with the adoption of CSCW and computer-mediated communication (CMC) systems: For any given individual, it not worth participating unless there is already a sizable group of people participating [14]. For communication systems and other CSCW systems with similar characteristics, this "critical mass" problem is often a barrier to starting up usage [23] [24].

The problem can be generalized. It is often the case that if use drops below a certain level, people will stop using a CSCW system. This "threshold effect" occurs because it is not worth even checking for activity if one believes that little will exist. The effect can be quite sudden, and often kills use of group communication systems. At the other extreme, people are attracted to systems where there is a considerable amount of activity, and this activity may excite or motivate them. Indeed, this social effect even exists in naturally occurring groups, as will be discussed below.

We have repeatedly encountered this problem in our own work. In field studies of one CSCW system, Answer Garden, users needed to be convinced that there was sufficient system activity by other users before they would bother to use it themselves [1]. Cafe/Espresso, a synchronous CSCW application that will be discussed at length below, ground to a halt after extended use in a field study because of this threshold effect. These experiences led us to consider what user interface mechanisms could provide a motivating sense of social activity on a CSCW system.

Why social indicators are important for CSCW systems

Before discussing related technical work, it will be useful to trace through some of the reasons that people might need some indication of social activity in their user interface. We draw primarily on the social psychology work on small group interactions.

People clearly use other people's visible activities in framing their own goals, motivations, and actions. As early as the 1890s, the social psychology literature had already discussed a "social facilitation" effect. This social facilitation effect is almost assumed within social psychology. As a standard textbook notes:

> ...we may therefore conclude that under either competitive or cooperative conditions group members working on a common problem communicate to one another a sense of urgency that tends to heighten their mobilization of energy, and thus their motivation. ([27], p. 467)

This effect has been found to be robust for face-to-face interactions, even those where people are working side-by-side silently. These findings are very telling: *Just knowing that others are doing some activity has a dramatic effect.*

This effect is not completely rational. Groups have been shown to have extra-rational and powerful effects on people's perceptions, decisions, and actions. For example, in a classic 1950s social psychology study, Asch showed that many experimental subjects will actually come to misperceive stimuli when surrounded by people reporting different results [3]. This effect was shown to be true, although weakened, even under conditions of anonymity [9]. There are many other instances of this effect, ranging from the bystander effect (i.e., an individual is much less likely to go to the rescue if no one else does) to contagion effects (e.g., running in a bomb scare or in a riot).

Unfortunately, the grounds for the social facilitation effect are not well understood. (There is a noticeable lack of theorizing in social psychology [25] [6].) Possibly, as Olson [28] argues, humans have an implicit game-theoretic view of the benefits and costs that occur in group interaction. McGrath [25], on the other hand, argues that people intrinsically need social learning and feedback.

In general, however, we can certainly say that people pay a great deal of attention to the activities of others. As McGrath points out:

> The point to be made here is that the presence and behavior of other people (and, in a sense, the absence and the inactivity of other people) help to define the meaning of situations for the individual and can have a powerful impact on his or her behavior, attitudes, and feelings in those situations ([25], p. 237).

We are not arguing that shared representations should be ignored as an interface concern. Clearly shared representations, when appropriate, are extremely important. Hutchins' recent work on distributed cognition (e.g., [18]) argues for the utility of shared representations. For example, airline pilots are able to partially put aside the details of jointly flying a plane because of their shared representation in the dials and controls [16]. As Hutchins points out, people will come to innovate and fit their social processes to the representations that are available [17].

However, we are arguing that there is a range of social indications that could be useful. Some systems may merely need general indications of activity -- perhaps just that activity is occurring. Most CMC systems fall into this category. Other CSCW systems may require further representations, particularly if the users know the representations are being shared with others. (Representations of general activity can themselves be a shared representation. However, when we use the term "shared representations" below, we will mean representations with a greater level of detail.)

Accordingly, user interface components that show part of this range of social indication -- social activity, shared representations, or some other social feature -- could be quite useful for CSCW applications. We have discussed how powerful these social effects can be, and we would like

to use them within CSCW systems to motivate system use and group performance.

Previous and related work

Indications of social activity were previously discussed by Dourish and Bly [12], Beaudouin-Lafon and Karsenty [5], and Dourish and Bellotti [11]. Each discusses their interests in terms of "awareness." Awareness is conceptually similar, but not identical, to our interest in social indication.

In Dourish and Bly's shared-media system, snapshots of users were provided to other users. Dourish and Bly argued that these images provided a sense of awareness to other users. In this sense, users could form images of other people's activities by directly observing them. A similar mechanism exists in Watcher [22] and Montage [33], as well as other synchronous audio and video systems. This service indicates social activity to its users and appears to be quite valuable to them. However, this particular visualization of activity is limited to particular types of systems, and many forms of system activity cannot be adequately addressed through this mechanism.

Beaudouin-Lafon and Karsenty use awareness differently, to connote the shared representation, through a shared workspace, in their system GroupDesign. They provide the user with a number of sharing capabilities, including being able to see what other users are examining and identifying the modification history of objects. Awareness for Beaudouin-Lafon and Karsenty also echoes the general "awareness" about shared-window system interfaces by Lauwers and Lantz [20]. Beaudouin-Lafon and Karsenty's awareness, then, aggregates what we see as important distinctions, including awareness of general activity, shared (and detailed) representations, and system-level requirements. It will be useful to analytically separate these types of awareness and to consider different indicators for each type.

Dourish and Bellotti define awareness as "an understanding of the activities of others, which provides a context for your own activity" ([11], p. 107). Since their interest is in group editing facilities, they promote the shared representations that are possible with a shared workspace. Other possibilities for social indication are noted, but are dismissed with the requirement that they need be explicitly generated or are restricted to users performing specific social roles.

We have seen in the previous theoretical overview that interfaces can provide a variety of social indications, ranging from general activity through shared representations.

Figure 1 shows the various types of indicators. (Consider Figure 1 as a tree with the root at the top; each circle is another type of indicator.) Using this diagram, we wish to make several points. First, all of these indicators and awareness mechanisms are visualizations of system and user states; hence, we consider them all visualization problems.

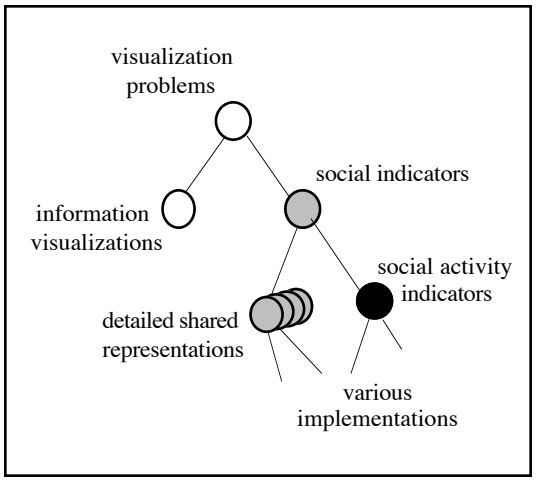


Figure 1: Social Indicators and Visualization User Interfaces

There has recently been substantial work in information visualization (e.g., [21]). While social activity is a form of information, we consider social indication to be a different, although related, problem. Information visualizers tend to be best for browsing or retrieving. Social indicators, especially those denoting general activity, can be used as alerting and control devices in the user interface, depending on real-time or near real-time performance. CSCW systems, especially synchronous CMC systems, do change significantly in real-time. However, we do note that at the extreme (e.g., process diagrams or shared information spaces), some detailed formulations of social activity do turn into information visualization problems.

Second, within the visualization of social settings, there are several types of social indicators, as Figure 1 shows. An important type indicates general social activity, as we have been discussing. Another type consists of shared representations, which themselves can range in the level of detail and the type of representation (e. g., whether user profiles or process models are included). There may be other social indicators as well. The term "awareness" as previously used in the literature fuzzily connotes all of these social indicators. We use a different term, then, to separate these concerns analytically, since using "awareness" is now loaded in differing ways.

Finally, we note that the problem is not specific to our system per se. We will be describing a system in a particular CSCW toolkit, the Cafe ConstructionKit. However, the interface mechanism could have been built in a number of other CSCW systems, including Suite [10], Rendezvous [15], and ConversationBuilder [19].

In summary, we have noted that providing user interface components for indicating social activity and social representations would be useful and powerful. We have also traced through the "awareness" literature and noted the various types of social indication. Having demonstrated some usefulness in these interface components, we next

demonstrate one mechanism for constructing these interfaces.

THE CAFE CONSTRUCTIONKIT

Our interface mechanism for social indication is constructed in a new CSCW toolkit, the Cafe ConstructionKit. In fact, the mechanism was constructed primarily because a specific application in that toolkit, a synchronous chat application called Cafe/Espresso, required it.

In order to explain our interface mechanism, we must first present the Cafe ConstructionKit. The Cafe ConstructionKit provides critical facilities and building blocks for the indicator mechanism. We will then briefly describe the Cafe/Espresso application.

The CafeCK toolkit

The Cafe ConstructionKit (CafeCK) is a CSCW toolkit for supporting the easy construction of applications such as information filters, locator services, digital libraries, and other CSCW projects. It can be used to add a variety of CMC and information components, such as information retrieval mechanisms, email readers, bulletin boards, synchronous talk, and socially constructed spaces, to many applications. In short, CafeCK provides a toolkit for sociality and information use.

To do this, CafeCK provides a set of reusable objects that include message transport for asynchronous and synchronous communication, parsing for a variety of semi-structured protocols, private and public channels for narrowcast communication, message filters, and message retrieval by a variety of semi-structured methods. CafeCK, based in C++, the X Window System, and the Xt toolkit, is programmable through the Tcl programming language [29]. Essentially, Tcl serves as "glue" between the computational objects (Figure 2); each object exports a number of Tcl verbs that allow it to be used by an application writer. By selecting from the set of available components (or by extending it) and by writing a simple Tcl program, an application writer can create a set of distributed processes to handle information retrieval, information access, or electronic communications. By configuring the objects and providing the suitable Tcl program, *any* application can include the functionality of bulletin boards, chat systems, and electronic mail filters. Additionally, we are actively working on providing the important construction facilities of Multi-User Discussions [7], so that users can interactively and collectively construct information access methods and environments on their own.

Because of this emphasis on providing CMC and information building blocks, CafeCK can provide a range of social functionality to CSCW applications. The architecture is open, unlike many information retrieval systems. It can also serve as a platform for testing various heuristics for interactive information seeking, where users work together to find, create, maintain, and store new information and knowledge.

We are designing CafeCK iteratively, and we are committed to design based on observing how people actually use the system. We are currently on the third version of CafeCK, based on our field studies of use. The first version showed that it was possible to provide a flexible, distributed construction set for interactive communications. However, informal user studies of the first version argued for a better command language (hence, Tcl), using a standard synchronous protocol (hence, the use of NCSA's Data Transfer Mechanism), user interface support for the interactive communication objects, and careful attention to scalability issues. The second version had a limited field study that consisted of a single application used over 2 months in a group of 14 users. We found that using Tcl allowed application and end-user programmers to create new applications without needing to radically reconstruct the CafeCK components, but we also confronted new requirements concerning performance and on-line maintenance issues. The current version, which addresses the issues raised in the field test as well as adding many new components, is under construction. The interfaces presented here were built using the second version.

CafeCK was designed to allow the easy construction of CSCW applications. One such application, with which we encountered the need for social activity indicators, was

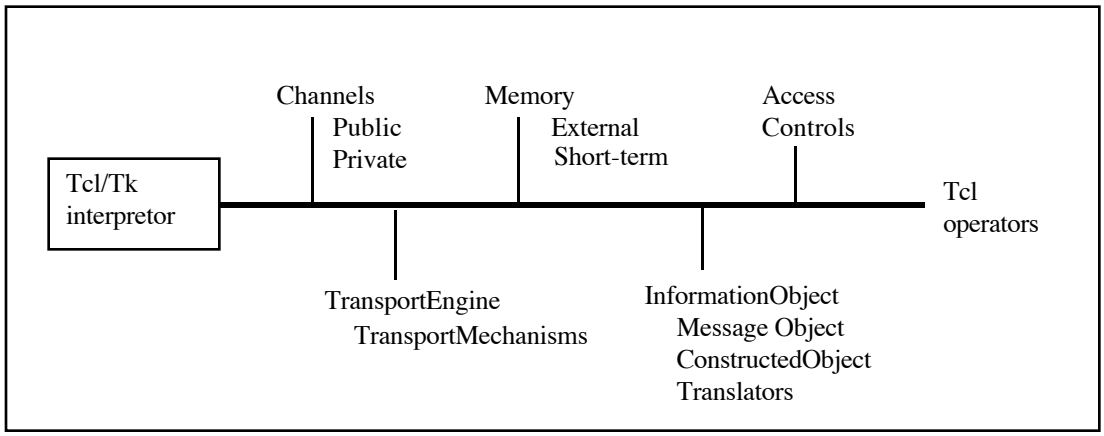


Figure 2: Tcl as the "glue" mechanism in CafeCK

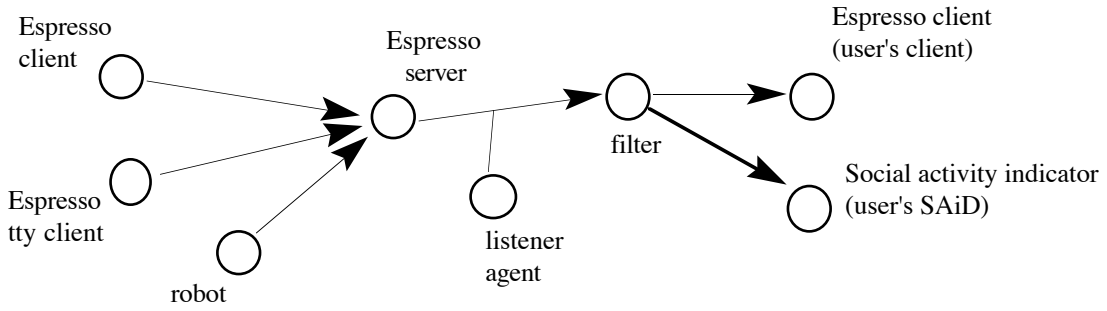


Figure 3: Espresso application architecture (sample)

Cafe/Espresso. This application is considered in the following section.

The Espresso application

Cafe/Espresso (Espresso) is a synchronous chat application constructed in CafeCK. It is functionally based on an existing system, Zephyr [8]. Figure 3 shows a sample configuration for Espresso. X-based and tty-based clients communicate with an Espresso server, currently using the NCSA Data Transfer Mechanism. Also, as shown on Figure 3, robot services can post; for example, we have one such robot for providing weather reports. The Espresso server broadcasts or narrowcasts the message according to the user's wishes.

Additional services can be put at will on the message stream. In Figure 3, the user has placed a message filter on the stream. The user's client (i.e., his reader) follows the message stream from the filter. The last service, the user's social activity indicator (SAiD), will be described below. As well in the Figure, there is a listener/agent on the stream; for example, one such listener checks for interesting messages to archive. This merely requires adding a database object to the pre-existing communication objects.

Users can type in messages of any length, although the current interface makes messages of less than 20 lines more probable. The messages can be sent to individuals, distribution lists, or broadcast channels. New channels can be added on an ad-hoc basis.

We built Espresso in order to examine lightweight communication mechanisms for exchanging information, asking questions, and receiving answers. (This is part of sequent work on Answer Garden [1, 2].) As mentioned, we examined Espresso usage in a two-month field study.

We hoped that Espresso would serve as an easy-to-use, facility. If you were present, you could ask a question or answer back. If you were not, the messages on the channel rolled by and were forgotten by your client. Users could ask their question, and if anyone had an answer and was paying attention, they could answer quickly.

However, after several weeks of use, Espresso encountered a standard CSCW problem. Espresso consumed screen real estate, and screen space is always scarce. Users therefore iconified Espresso. But, once Espresso was iconified, no one could see that new messages had arrived. Once this began, others knew that Espresso windows were iconified. They knew they would be unlikely to get answers, and they never bothered to ask their questions. Once there were no questions, no one bothered to de-iconify their application and look for new questions. Thus, the number of active users fell below the necessary threshold, and use plummeted.

In the Espresso application, we needed some visual mechanism that indicated when messages arrived from interesting people. In other words, we needed a mechanism to add a social activity indicator, some means of knowing that there was system activity.

Restricting the social indication problem

While many CSCW systems need some type of shared representation, Espresso users needed only an indicator of system activity. A shared representation of the message content was not necessary. In fact, for large-scale use (e.g., hundreds of users), providing some shared representation might have been counter-productive because of the potential for information overload.

Therefore, we concentrated on social activity indicators. We did this for several reasons:

- They can be generalized to many different CSCW systems.
- Within a particular domain of interest, CMC systems, shared representations have little meaning. Social activity indicators are therefore more valuable for this important class of CSCW systems.

The building blocks in the Cafe ConstructionKit permitted us to easily construct the social activity indicators we needed. To do so, we constructed a new service in CafeCK.

SOCIAL ACTIVITY INDICATORS

Espresso, as explained above, was vulnerable to both the critical-mass problem and threshold effect. Espresso required that people pay attention to the application, but people were unwilling to pay attention unless others were using it.

To ameliorate this problem, we devised a CafeCK service to indicate social activity on the system. We called this service the Social Activity Indicator/Display (SAiD). In Figure 3, SAiD appears as a stand-alone process, with separate windows on the user's screen.

In general, the functionality of SAiD is contained in three subcomponents (Figure 4). Within SAiD, messages flow through components that filter the messages, analyze the message traffic for a variety of measures, and then draw the activity indicators.

Before any of this can happen, however, the user must define one or more groupings (such as channels or users) to track and depict. Additionally, for more complex graphs, the user can assign a prototypical message or a set of terms to each grouping, representing topic areas of particular interest to him.

Because of the nature of CafeCK, filtering can be either external or internal to SAiD. We allowed this extra layer of filtering as a convenience to the user; CafeCK's building block approach allows filters within any service. Figure 4 shows the filtering being done within SAiD.

After filtering, each incoming message is first examined by SAiD's analysis engine. Currently, analysis consists of three steps. As messages arrive, they are matched to user-created rules that place them within the user groupings. These rules are based on their semi-structured fields (i.e., any system or user defined header field). Rules may use field values, calculate new fields, or logically test fields. For example, one could have a rule to watch for particular users on particular channels:

```
include-message
  ((member($this.From, member-list) &&
    member($this.Channel, channel-list)),
   group1)
```

The rule syntax, like that for the CafeCK "glue" language that binds the building blocks together, is just syntactic sugar for Tcl interpretation. In the above rule, member-list and channel-list have been defined as internal lists of values or as external files. The variable "$this.From" uses CafeCK syntax for indicating the From field of the current message. Rules may also include calculations:

```
include-message
  ((minute($this.Date) >=
      minute($Today) - 5))
```

SAiD's analysis engine then calculates activity measures for each grouping. Currently we use two simple measures, message length and number of messages. Because SAiD is extensible in the same way as any CafeCK service, new activity measures (or any other type of measure) can be added to the analysis engine.

Finally, the analysis engine analyzes semantic content and creates a normalized word count vector [4]. The semantic relevance of each user grouping is calculated by comparing it to the prototypical message or set of terms for that grouping. The dot product is used as a measure of relevance.

The result of these analyses is used to update the corresponding activity and semantic information of each user grouping. The update process currently assigns a weight to previous and present activity within each grouping, allowing one to place a premium on current activity. This weight, like most other settings, can be set by the user during run-time.

After analysis, SAiD then invokes its display engine. Within the display engine, one or more display objects can be assigned to each grouping. These display objects must be updated in real-time, and therefore fast updating is required. Each display object associated with a grouping is

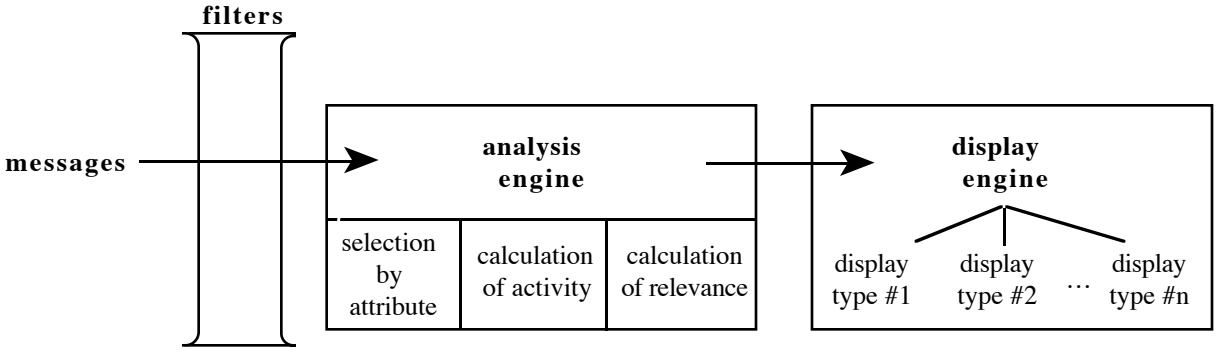


Figure 4: Social Activity Indicator/Display (SAiD) process

passed a vector consisting of the activity and relevance of each user grouping. We have found that each display object should keep little or no state, updating its display based solely on its current state and the new vector.

CafeCK provides important facilities for constructing SAiD. Because SAiD was built in CafeCK, SAiD required no additional communication, message handling, message filtering, or message parsing capabilities. The relevance calculation and rule traversal, while built first for SAiD, will be included as standard components within CafeCK. We were able to prototype these components in Tcl and then move them to C++. The display objects, however, did require specially constructed code in Tk (Tcl's user interface toolkit) and [incr Tcl] [26].

The "glue" code for the SAiD service is merely:

```
while (True)
  {
    CafeNextEvent($this);
    Said_Analysis($this,analysis_vector);
    Said_Display($this,analysis_vector);
  }
```

To add a new component, one merely adds (or modifies) this dispatch loop. Note that more complex analysis engines, different displays, or even other social indicators can be added easily.

Example social activity indicators

We have constructed three families of social activity indicators. We offer these as sample visualizations for indicating social activity.

Figure 5 shows two examples of the "simple indicator" family. Like the xbiff tool, which notifies the user of incoming email, this simple indicator merely registers the presence of incoming messages. In fact, one can think of this tool as a group-biff (or as we prefer to call it, biff.n.friends). Most of the analysis engine's results are not used.

In our prototype, the simple indicator shows a bitmap of a telephone on its cradle. A variety of options exist for this simple indicator. The bitmap can alternate with one of the phone ajar, the bitmap can flash, or the icon can jump about (within a narrowly confined area). In Figure 5(a), the bottom of the indicator shows the sender of the last message; it is assumed this is for one channel on the system. One can get a fair understanding of the channel activity by watching the speed at which the icon changes; the animation is critical to the user. Figure 5(b) shows a similar indicator; however, this one shows the channel and the number of messages within the last 5 minutes. We have also constructed a similar indicator that shows images of the message senders.

We have also constructed two more complex families of social activity indicators. We do not doubt others are possible. The first uses a bar chart to note the level of system use across channels, users, message types, or other groupings. Figures 6(a) and 6(b) show this display. (Note that these displays are normally in color.) It is difficult again to convey the animation in print, and the animation is critical to the user's perception. In these figures, channel activity is being graphed. Thus, in Figure 6(a), there are currently five active channels. Some time later (in this case, 5 minutes), activity on the various channels has changed. The display shows six active channels, and the activity within each has also changed.

Because we felt that the user's interest in past activity would be less than for current activity, the length of the bars decays over time. The decay setting (i.e., the time for which activity is remembered) is user settable since users and conditions vary.

Figure 6(c) is a social network diagram. It displays not only activity by channel, but shows the semantic relevance of the message content. Within the social network diagram, each circle is arrayed along an arc in a polar plot. The diameter of the circle represents the amount of group member activity, and the distance from the circle to the origin represents the relevance of the member. We plan to add the calculation of group similarities to this diagram (as in [30]); this would position the circles relative to one another.

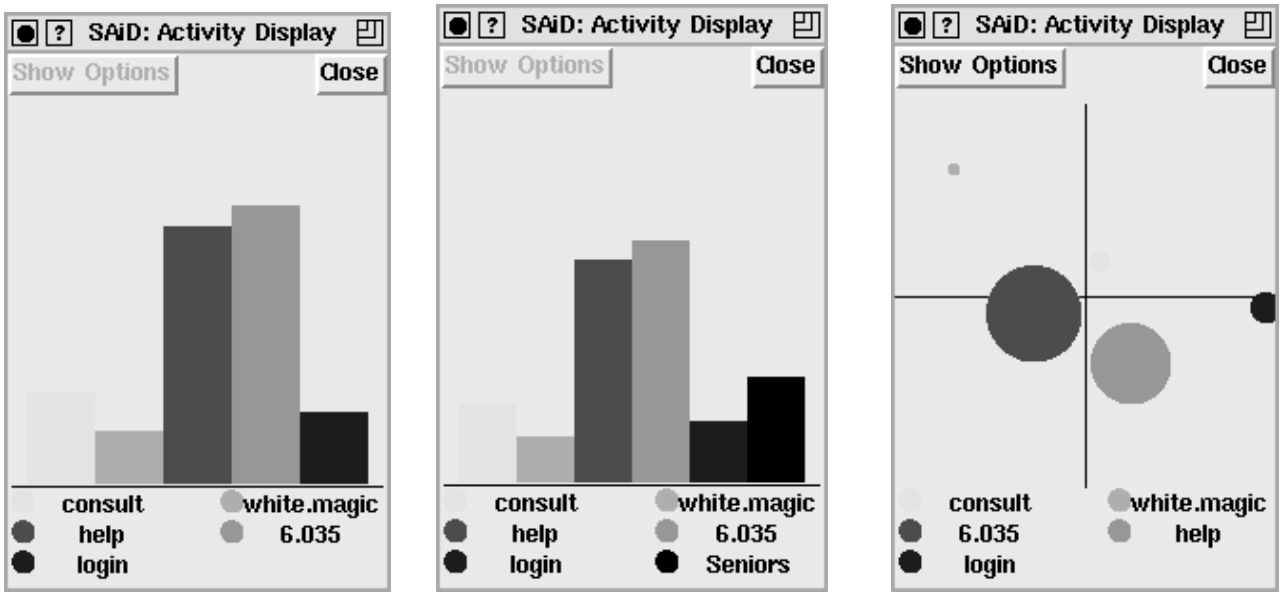
The display can apply a fisheye lens to either the relevance or activity of each visible object [13]. The fisheye effect is under the control of the user, and the introduced non-linearity of the graph is useful for emphasizing or de-emphasizing differences in values for groupings [31].

Each display is under the full control of the user using a direct manipulation interface. As mentioned, the user can set the groupings and the attributes for each grouping (e.g., message type or time). As well, the user can set the threshold values for each attribute, the threshold values for notification, magnification values, the fisheye effects, and the display types. Resetting the fisheye value is often

(a) The indicator displays the sender of the last received message

(b) The indicator displays the channel and number of messages within a fixed period of time.

Figure 5: Simple indicators (group-biff) for the "help" channel.



(a) Activity graph for five channels

(b) Activity graph for system system channels at a later time

(c) Social network graph for five channels

Figure 6: Various SAiD displays. The groupings are normally in color.

valuable in analyzing complex or heavy system activity without having to read the message traffic.

Informal lessons

We have informally tested our social activity indicators using a data set consisting of 6559 messages on 14 channels. The most difficult part of examining these indicators has been the need for naturalistic data. We were fortunate to obtain two days of data from another chat-like system. We also created a robot that can take a log file (or any other source of data) and feed it into a CafeCK application.

Because these are real data, these messages show the characteristic temporal flows that one finds in a synchronous communication system. There are bursts of frantic activity, often followed by long lulls. Because of the amount of message traffic and channels, we believe that this data is a realistic environment for examining these indicators.

We found:

- The simple indicators are best when the user wishes to watch a single channel. They are easy to comprehend and consume little screen real estate. However, they are not sufficient for complex flows of information or for many channels. While one can have a bank of these indicators (perhaps each one watching a different channel), it quickly becomes confusing. Additionally, with our current architecture, multiple simple indicators require multiple SAiD processes.
- The bar charts adequately represent activity on less than ten channels or other groupings of interest. The simplicity seems to be key. A display using a 5x5 grid (with activity shown through color) was difficult to interpret. While bar charts are limited in the types of information they can convey, they are surprisingly useful in synchronous systems.
- We believe that the social network diagrams offer the most potential for representing complex states with multiple indications of activity. However, these diagrams also require a cognitive effort to learn and interpret, perhaps limiting their usefulness with the general population.

In general, we found that having a building-block approach was extremely useful in constructing CSCW services. It would have been difficult, at best, to construct SAiD without the CafeCK message facilities.

CONCLUSION AND FUTURE WORK

We began with the premise that people wanted to know what was going on about them, and that capability should be reflected within interface design for CSCW applications. In this paper, we have shown that building a generalized capability for indicating social activity is quite possible.

Moreover, within this paper, we showed what CSCW capabilities were required for the easy construction of such interface mechanisms. Through the facilities of the Cafe ConstructionKit, we were able to easily construct several families of social activity indicators.

Many extensions to this work exist. One such extension is to examine the interplay between shared representations and activity indicators. We are currently designing activity indicators and a process diagram to allow editors, reviewers, and authors to track papers for a journal. While editors may need a shared representation, they may prefer to restrict authors to only activity information. Nonetheless, this activity information might be quite valuable and useful. As process engineering becomes even more prevalent, it will be important to consider the interface requirements for making these process diagrams *socially* usable and useful.

Additionally, we would like to examine the use of overlays in social indicators [32]. Users could control the amount of information in the social activity indicators by adding or removing overlays. This may prevent information overload, allowing us to create more complex social activity indicators.

Acknowledgments

This project has been partially funded by grants from the UCI Committee on Research and NASA (NRA-93-OSSA-09). This work began with a remark in 1989 when Wendy Mackay noted that one should show glass boxes rather than black boxes in the interfaces of distributed systems. This project also benefited greatly from conversations with Jonathan Grudin, Debby Hindus, Eric Mandel, Paul Dourish, and Lynne Markus. The other members of the CafeCK research group, David McDonald, Andy Tipple and Jeff Dorsz, contributed to this understanding of social indicators. Anna Krylovesky and Iman Ozgur implemented the simple indicator, and Jim Whitehead ran the Espresso field study.

References

1. Ackerman, M. S. Answer Garden: A Tool for Growing Organizational Memory. Massachusetts Institute of Technology, Ph.D. Thesis, 1993.

2. Ackerman, M. S. Augmenting the Organizational Memory: A Field Study of Answer Garden. *Proceedings of Conference on Computer-Supported Cooperative Work (CSCW'94)*, 1994: 243-252.

3. Asch, S. E. Effects of Group Pressure upon the Modification and Distortion of Judgments. In Proshansky, H. and B. Seidenberg (ed). *Basic Studies in Social Psychology*. Holt, Rinehart and Winston, New York, 1965.

4. Bartschi, M. An Overview of Information Retrieval Subjects. *IEEE Computer*, 1985, May 1985: 67-84.

5. Beaudouin-Lafon, M. and A. Karsenty. Transparency and Awareness in a Real-Time Groupware System. *Proceedings of Symposium on User Interface Software and Technology (UIST'92)*, 1992: 171-180.

6. Collier, G., H. L. Minton and G. Reynolds. *Currents of Thought in American Social Psychology*. Oxford, New York, 1991.

7. Curtis, P. and D. A. Nichols. MUDs Grow Up: Social Virtual Reality in the Real World. Xerox PARC, manuscript, 1993.

8. DellaFera, C. A., M. W. Eichin, R. S. French, D. C. Jedlinsky, J. T. Kohl and W. E. Sommerfeld. The Zephyr Notification Service. *Proceedings of Winter 1988 Usenix Technical Conference*, 1988: 213-220.

9. Deutsch, M. and H. B. Gerard. A Study of Normative and Informational Social Influences upon Individual Judgment. In Proshansky, H. and B. Seidenberg (ed). *Basic Studies in Social Psychology*. Holt, Rinehart and Winston, New York, 1965.

10. Dewan, P. and R. Choudhary. A High-Level and Flexible Framework for Implementing Multiuser User Interfaces. *ACM Transactions on Information Systems*, 1992, 10(4): 345-380.

11. Dourish, P. and V. Bellotti. Awareness and Coordination in Shared Workspaces. *Proceedings of Conference on Computer-Supported Cooperative Work (CSCW'92)*, 1992: 107-114.

12. Dourish, P. and S. Bly. Portholes: Supporting Awareness in a Distributed Work Group. *Proceedings of ACM CHI'92 Conference on Human Factors in Computing Systems*, 1992: 541-547.

13. Furnas, G. W. Generalized Fisheye Views. *Proceedings of CHI '86*, 1986: 16-23.

14. Grudin, J. Why groupware applications fail: problems in design and evaluation. *Office: Technology and People*, 1989, 4(3): 245-264.

15. Hill, R. D., T. Brinck, S. Rohall, J. F. Patterson and W. Wilner. The Rendezvous Architecture and Language for Constructing Multiuser Applications. *ACM Transactions on Computer-Human Interaction*, 1994, 1(2): 81-125.

16. Hutchins, E. How a Cockpit Remembers Its Speeds. Manuscript, 1991.

17. Hutchins, E. Organizing Work by Adaptation. *Organization Science*, 1991, 2(1): 14-39.

18. Hutchins, E. *Cognition in the Wild*. MIT Press, Cambridge, MA, 1995.

19. Kaplan, S. M., W. J. Tolone, D. P. Bogia and C. Bignoli. Flexible, Active Support for Collaborative Work with ConversationBuilder. *Proceedings of Proceedings of ACM CSCW'92 Conference on Computer-Supported Cooperative Work*, 1992: 378-385.

20. Lauwers, J. C. and K. A. Lantz. Collaboration Awareness in Support of Collaboration Transparency:

Requirements for the Next Generation of Shared Window Systems. *Proceedings of ACM CHI'90 Conference on Human Factors in Computing Systems*, 1990: 303-311.

21. Mackinlay, J. D., G. G. Robertson and R. DeLine. Developing Calendar Visualizers for the Information Visualizer. *Proceedings of User Interface Software and Technology (UIST'94)*, 1994: 109-118.

22. Manandhar, S. Activity Server: You Can Run But You Can't Hide. *Proceedings of Usenix Summer Conference*, 1991: 299-311.

23. Markus, M. L. Toward a "Critical Mass" Theory of Interactive Media. In Fulk, J. and C. Steinfield (ed). *Organizations and Communication Technology*. Sage, Newbury Park, CA, 1990.

24. Markus, M. L. and T. Connolly. Why CSCW Applications Fail: Problems in the Adoption of Interdependent Work Tools. *Proceedings of Computer Supported Cooperative Work (CSCW'90)*, 1990: 371-380.

25. McGrath, J. E. *Groups: Interaction and Performance*. Prentice-Hall, Englewood Cliffs, NJ, 1984.

26. McLennan, M. J. [incr tcl] - Object-Oriented Programming in TCL. AT&T Bell Laboratories, manuscript, 1994.

27. Newcomb, T. M., R. H. Turner and P. E. Converse. *Social Psychology*. Holt, Reinhart and Winston, New York, 1965.

28. Olson, M. *The Logic of Collective Action*. Harvard University Press, Cambridge, MA, 1965.

29. Ousterhout, J. K. *Tcl and the Tk Toolkit*. Addison-Wesley, Reading, MA, 1994.

30. Romney, A. K. and S. C. Weller. Predicting Informant Accuracy from Patterns of Recall among Informants. *Social Networks*, 1984, 6: 59-77.

31. Sarkar, M. and M. H. Brown. Graphical Fisheye Views. *Communications of the ACM*, 1994, 37(12): 73-84.

32. Stone, M. C., K. Fishkin and E. A. Bier. The Movable Filter as User Interface Tool. *Proceedings of ACM Human Factors in Computing Systems (CHI'94)*, 1994: 306-312.

33. Tang, J. C., E. A. Isaacs and M. Rua. Supporting Distributed Groups with a Montage of Lightweight Interactions. *Proceedings of ACM Conference on Computer-Supported Cooperative Work (CSCW'94)*, 1994: 23-34.