Registration for Real-Time Groupware

Mark Roseman

Integrated Systems Applications Corp.
Suite 835, 10040 – 104 Street
Edmonton, Alberta, Canada T5J 0Z2
(403) 420-8081 roseman@edm.isac.ca

Saul Greenberg

Department of Computer Science
University of Calgary
Calgary, Alberta, Canada T2N 1N4
(403) 220-6015 saul@cpsc.ucalgary.ca

ABSTRACT
This paper examines the groupware registration problems, both social and technical, that occur as users select their tools and the others who will share them. We argue that users require far better and more flexible registration than is provided in today's systems. From our examination of users' needs we derive a set of requirements for groupware registration tools. We present a general registration architecture for creating different registration scenarios, and illustrate a number of the scenarios that we have created.

KEYWORDS
real-time groupware, registration, pre-meeting process, flexibility, toolkits, architectures

INTRODUCTION

Collaborations exist well beyond the boundaries supported by today's real-time groupware systems. In this paper, we address one of the many activities peripheral to the main groupware task, *groupware registration*. Registration is the process by which users of groupware select the tools and other users they want to work with. All real-time groupware therefore supports some form of registration. But is the level of support sufficient to satisfy the needs of groupware users? We would argue that it is not.

Our concern with registration has evolved out of work on our groupware toolkit, GroupKit [18]. A main goal has been to help developers build groupware applications flexible enough to accommodate the varying working styles of end user groups. Registration is one area that is highly sensitive to such variations. Though difficult enough for a single group using a single application, in our toolkit we need to provide good registration support for many groups across many applications.

This paper discusses the issues and problems that arise when building a system to support registration *in general*, not just for a particular group or system. A general registration system therefore allows constructing *specific* systems to suit different registration styles or scenarios. Though our motivation to study general registration comes from building a toolkit, this discussion will benefit developers of individual applications. After all, even a single application can be used by different groups and in different ways, and so may need to deal with multiple styles of registration.

The paper opens by investigating registration inside the pre-meeting process. We look at some related work on registration issues, and build the case that registration must be flexible enough to suit the social and work context of users. From there, we distill a set of requirements for a general groupware registration system. These requirements lead on to a discussion of the general registration architecture we have built into GroupKit. We then provide a number of examples of specific systems, built using this architecture, that were designed for particular registration scenarios. The paper concludes by raising a number of issues which need to be addressed to further refine our general registration architecture.

THE PRE-MEETING PROCESS

Groupware should support rather than disrupt existing social processes [9]. Consequently, software-based registration systems must consider the real life processes that people follow when getting together for a meeting. This section reviews the human factors in the *pre-meeting process*: what groups do when setting up meetings; how contact between group members is made or, in the case of ongoing work, re-established; and the inter-personal nuances that occur when people actually meet. Later, we will abstract these factors as a set of requirements for registration design in groupware.

Bostrom et. al. [1] see the meeting as a process that cycles through three phases: pre-meeting setup, during-meeting activities, and post-meeting teardown (which could lead into the next meeting). It is the pre-meeting phase that is relevant to registration, and the variety of activities occurring here are identified by Dubs and Hayne [7].

1. Setting goals:
   - review previous meetings to understand the status of the on-going process;

- describe meeting goals that establish a purpose to the meeting.

2. Getting participants:
   - develop roster of appropriate potential attendees;
   - inform participants of the meeting.
3. Collecting materials:
   - develop and gather necessary documents;
   - select and book equipment, such as presentation tools or process aids.

Points 2 and 3 together can be considered the key parts of an *active* registration process. Lists of people, equipment, tools, and documents are made; it is active because registration gathers the items together for the actual meeting.

The model above primarily addresses the planned meeting. As elegantly stated elsewhere, many meetings are quite informal and even spontaneous, the result of chance and one-person initiated encounters [10]. Cockburn and Greenberg [3] summarize two key factors that affects the ways people and groups get together in these situations. First, people must get in touch with the people they are interested in. They require a sense of who is around and how they can be reached, how particular people can be found, the social status of others as well as their willingness to be disturbed or interrupted. The second factor is that people need to choose appropriate communications channels. This is strongly influenced by the task at hand, what channels and tools are available to participants and what affordances they offer, the intended time period of interaction, the inertia involved in switching to a better communication channel, and how conversations can migrate across time and space boundaries. While people transparently handle these matters as part of normal social conventions, Cockburn and Greenberg argue that technology which is insensitive to these concerns often creates barriers that hinder or block our natural processes for handling them.

Louie, Mantei and Sellen [11] also present a model of communication activity that occurs when one person meets with another. Before a meeting begins, one finds participants in the *pre-communication* phase, attracts their attention in the *attention* phase, and then uses the *greeting* phase as a ritual for negotiating how communication will proceed. The main meetings communication occurs in the *maintenance* phase, followed by a highly structured agreement to end the session through a *closing* phase. The final *fade-out* ends the communication, and as the name implies, is rarely abrupt.

Is it worth pursuing a model for a "standard" pre-meeting procedure? While he doesn't address meeting phases explicitly, Grudin [9] argues that work processes can be described as "... the way things are supposed to work, and the way they do work." He highlights that fundamental components of most people's work activities are error and exception handling, personal improvisation to fit contextual considerations, shortcutting and modifying standard procedures. As both Suchman [22] and Grudin have pointed out, procedures are often *post hoc* rationalizations of *ad hoc* activities. In this light, we see the registration process as a highly variable and dynamic set of activities, influenced by personal desires, interpersonal relationships, and the situation at hand.

Registration is a very important part of the pre-meeting process, for it captures the essence of how people and their artifacts are brought together. Because groupware users are separated by distance, it is exactly this part of the pre-meeting process that must be handled by groupware. If the technology does not support a realistic registration process, then the entire meeting can be disrupted. To illustrate, consider any of the following pre-meeting situations. While laughable from a human point of view, poorly designed registration technology could make these very real in groupware.

- Strategic, highly political meetings of a large company are operated under an open door policy where absolutely any person could walk in and participate.
- A "town hall" meeting keeps its participants hidden, so no one really knows who is present.
- Casual hallway conversations cannot occur unless they are "authorized" by a company administrator.
- A pre-meeting process demands that people answer numerous questions, fill out long forms, and exhaustively state all meeting needs before they are actually allowed to get together.

In real life, collaborations take place within a variety of social and work contexts, highly dependent on the group and its situation. The registration process, whereby these collaborations are formed or reestablished, is very sensitive to these variations, and an inappropriate context can impede or disrupt the resulting collaboration. To cope with this, the real-life registration process is necessarily diverse, and accommodating to the situation at hand.

WHAT REGISTRATION MUST PROVIDE TO GROUPWARE USERS

We now turn our attention to how people's registration needs can be supported in groupware. Because of the inherent variability of the pre-meeting process, we firmly believe that registration must be tailored to the group, its needs, and its particular situation. Thus our view is not of a single registration process, but of one offering a spectrum of capabilities — perhaps through a suite of tools — that users can choose from.

This section describes the capabilities that a general groupware registration architecture should provide its users. The first point below addresses the fundamental role of registration. The second considers the information people generally need to support their own social roles in the registration process. The remaining points consider the nuances of how registration systems are matched to people, their working contexts, and the actual groupware applications.

Support registration of both people and their artifacts. Two key activities of the pre-meeting phase are selecting and collecting participants, equipment and meeting artifacts. In groupware, "collecting participants" involves helping people find who is available and what meetings they can join, and to establish appropriate communication channels between them. The "equipment" is the groupware applications people use, while the "artifacts" are usually the documents and other items manipulated through the application programs.

Support the distribution and access of registration information. Registration is not just a technical process of connecting software, but part of a rich social activity. For people to fill their personal roles and make registration decisions, they require cues about their environment. These cues include such things as who is around, their current activities, and whether or not they can be interrupted. This information should be accessible through the system in a variety of ways, and its distribution — depending on the context — may be controlled or limited by the group (e.g. issues of privacy or secrecy). The information should help participants answer a host of questions particular to their pre-meeting needs.

Support a spectrum of group involvement in registration decisions. Different situations demand different amounts of group involvement in registration. The issue of who can join an existing group may be decided in a number of ways: a single group member such as a facilitator may decide, the group as a whole may vote, come to a consensus decision, or even ignore the issue entirely, by letting all newcomers join. For informal interaction, control over accessibility, privacy and interruptability is the issue. Groupware registration should give groups the means to make their own registration decisions.

Support users' individual needs for registration. People may have different roles and preferences in meetings. Registration can acknowledge this "individual context" by having providing components tailored to the needs of its participants. Differences may be relatively cosmetic, such as the same registration policy presented with a different interface style. Perhaps the components vary the detail of information presented to match what a participant wants to know. Deep differences occur when people assume roles with special capabilities, such as when one person holds control over who can actually join a meeting.

The same registration style should be usable with different groupware tools. Registration sets the tone and context in which people gather and tools are used. Several groupware applications often exist within this same context. As we do with real world counterparts, we should be able to use the same registration policies and conventions to access a wide variety of meeting tools such as whiteboards and flip charts. Allowing the use of the same registration style across groupware applications also makes it easier for users to leverage their knowledge when learning new tools.

Different registration styles should be usable with the same groupware tool. Conversely, the same groupware tool can be used in a variety of ways, such as a drawing tool being used for brainstorming, engineering designs, leaving notes for people, or more frivolous pursuits. Each of these suggests a different usage context for the tool, and a different way of gathering people together to use it. As we have argued, different registration styles can either aid or hinder the effective use of a tool in a given context. People should be able to access tools with the appropriate registration style.

Support dynamic registration, so that people and artifacts can be added or removed at any time. In real life, people enter meetings at quite different times. For stereotypical structured meetings, we imagine that the list of participants and resources is known ahead of time, and all attendees arrive on time and leave when the meeting is formally closed. In practice, people drop in uninvited (but may be welcome anyway), arrive late, and leave early. In informal interaction, the "list" of attendees is defined by the actual encounter. Artifacts can be brought in at various stages of the meeting. Registration schemes must be flexible enough to allow people to enter meetings, and bring in new resources, at any time. As a result, registration is actually an on-going process, continuing from the pre-meeting setup to the during-meeting activities and post-meeting teardown.

REGISTRATION IN CURRENT SYSTEMS

This section briefly reviews the state of registration in today's groupware applications. To date, most groupware has focused on supporting the group's needs only within the confines of an established groupware session. Registration, though necessary to establish these sessions, has played a minor part in today's systems, providing only the minimal capabilities needed to create communications channels. Registration has generally not received the attention we argue it deserves. Still, several good examples do exist, particularly for specific registration scenarios. By surveying implementations within the context of the previous requirements, we can identify common strategies, suggesting reasons why some work, and why others fail.

At the CompCon conference in the late 1960's, Doug Engelbart revolutionized people's conceptions of computers when he presented his Augment system [8]. One of its capabilities was shared view conferencing, and during his presentation we saw him initialize the session by asking his partner (over a voice/video link) the physical number of his terminal. Registration was the act of supplying the magic number to the system. Twenty-five years later, many systems still provide only the most rudimentary support for registration and connectivity between groupware users. Instead of terminal numbers, systems now often require users to identify groupware sessions by low-level addressing such as the Internet host name and TCP/IP port number. An example is NCSA's Collage illustrated in Figure 1 [12]. Slightly higher level

Figure 1. Registration in Collage.

are services where one supplies their partner's userid and host name. For example**,** the Unix text-based chat facility is initiated with the command

talk <userid>@<machine name>.

Low level registration fails to pass many of the criteria listed above. It is inflexible, and the information required by the system may not be readily available to the user.

Some systems do not allow latecomers to sessions, insisting instead that all users are identified at startup, e.g. versions of Rendezvous [16] and MMConf [4]. This restriction greatly simplifies the construction of systems. Designers do not have to worry about the mechanism for bringing the displays of new users up to date, and handling of communication links is easier. However, this restriction is at odds with the way people actually join meetings.

This is not to say that all systems that are simple to implement are inappropriate or unusable. For example, *common rendezvous points* allow users to register simply by establishing a communication connection to a central location. To illustrate, users of text-based Multi-User Dungeon (MUD) systems join an on-going chat and game session by connecting to a well-known Internet address via telnet [5]. The prevailing policy of any user being allowed to enter the MUD is compatible with their essentially recreational nature. The connection interface is straightforward and understandable to MUD users, and the address of the particular MUD site reasonably easy to note down; registration works well for the context it aims to support.

There are two classes of systems where registration has been reasonably successful, by designing it to support particular styles of meetings. First are systems that support casual interaction. These are usually centered around a media space of some sort (e.g. Cruiser [17], Cavecat [11], Portholes [6]) but can also rely on more conventional technologies (e.g. Telefreek [3]). These systems are highly tuned to the social contexts in which they are embedded, taking into account privacy, awareness of other users and their activities, and preserving the social cues people use when collaborating. Some systems leverage people's conceptual models of casual interaction by providing a variety of registration metaphors e.g. person-centered, space-centered, and time-centered [11].

Facilitated group support systems are the second class of systems fitted to a particular meeting style [13]. Because they are designed for executives who do not have time to deal with the quirks of technology, registration is moderated by a trained facilitator who performs much of the clerical duties for the pre-meeting setup. As with casual interaction, the scheme works well for a particular meeting style but does not necessarily generalize to others.

The one initiative that we are aware of that seriously considers *general* registration (rather than particular styles such as casual interaction support or facilitated group support systems) is the "mmusic" working group of the Internet Engineering Task Force. Their goal is to produce a "multiparty multimedia session control" protocol that will provide registration support to Internet-based groupware and multimedia conferencing tools [20, 21]. The protocol is being designed to acknowledge the variability inherent in registration styles and overall session policy. Issues covered include late joiners, floor control, interaction style, and mechanisms to negotiate and change session policies. What is interesting is that they are using a range of meeting and communication scenarios to elicit the variability of registration styles, as illustrated in Figure 2.

In summary, there has been scattered success in developing context-specific registration mechanisms. Notions of general registration are still in very early stages.

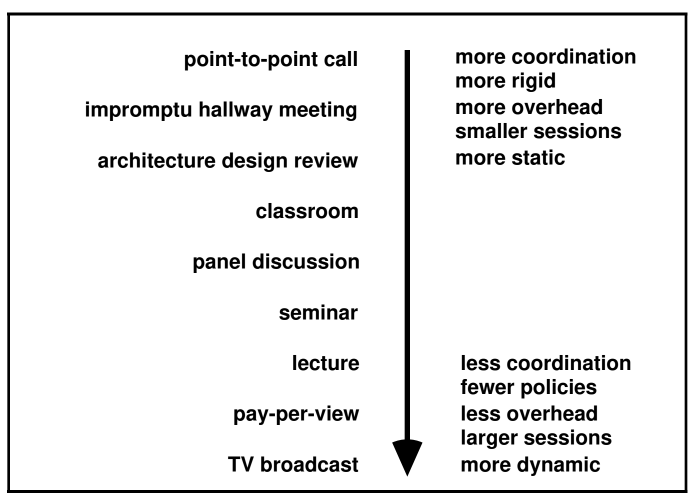

Figure 2. Usage scenarios investigated in mmusic. Reproduced from Schooler and Weinrib [21].

A REGISTRATION ARCHITECTURE

We have built a groupware toolkit called GroupKit to provide developers with an infrastructure for implementing real-time groupware systems [18]. One of our key concerns was how to design GroupKit's architecture to cope with the diverse registration schemes required by groupware developers, ultimately driven by the needs of their users. The architecture has proven successful, both in letting us build a variety of registration prototypes, and in providing a better understanding of general registration.

This section begins with a snapshot of GroupKit's registration architecture and its features. The power of this architecture is then demonstrated by illustrating a broad range of prototype registration systems we have constructed with it.

Features of the Registration Architecture

A brief overview of GroupKit's run-time architecture is necessary to understand how registration is handled. Further details are found in Roseman and Greenberg [18].

GroupKit user sessions consist of a number of mostly replicated processes arranged across a number of machines. All processes maintain communication facilities necessary for exchanging messages with each other. Figure 3 illustrates an example of the processes running when two people (George and Mary) are communicating to each other through two groupware applications 'A' and 'B' (perhaps a group sketchpad and editor). Processes are represented by ovals, and the lines joining them indicate communication paths. Each person would see windows representing each application on their screen. They also have a window representing their *Registrar Clients* — the interface and policy supplier of their registration system.

The actual components work as follows. The central *Registrar* maintains a list of all the conferences active on the system and a list of all the conference users. The Registrar itself does not decide how conferences are created or deleted nor on how users join or leave conferences. The *Registrar Client* (one per user) allows users to create, delete, join or leave conferences. It interacts with other Registrar Clients through the central list provided by the Registrar. The Registrar Client provides both a user interface as well as a policy dictating how conferences are created or deleted and how users are to enter and leave conferences. Different Registrar Clients can be created to suit different registration needs. Finally, the replicated conference application is the actual shared application built using GroupKit, and is separate from the registration system.

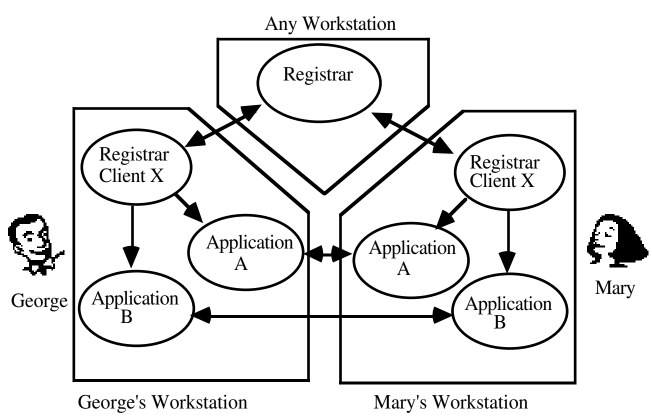

Figure 3. Basic run-time architecture of GroupKit.

The important architectural features of the registration components are described below. These features, as well as the examples later on, will indicate how GroupKit can satisfy our registration requirements.

Decouple Applications from Registration. A key feature of our architecture is that the specific groupware applications are decoupled from the registration process itself (the Registrar Clients). Different groups can use different Registrar Clients and thus different registration policies. Participants within a single session can also have their own special Clients (that coordinate their policy with the other Clients in that session). Decoupling explicitly acknowledges that no one registration scheme can deal with all the registration needs its users may have, and that registration need not be tied to a particular application. This allows us to build, use, and substitute a wide variety of registration schemes with a single groupware application, and also to use the same scheme with different applications. It achieves the flexibility necessary to combine registration systems and applications based on users needs, not technological restrictions. An additional and important benefit is that writing application code is greatly simplified, since the registration component is handled separately.

Standardize Communication Mechanisms Between Application and Registration System. Despite being decoupled, the application and registration components must communicate so that, for example, the application knows when a new user enters the group. Remember that applications do not know details of the particular registration system they are attached to (and vice versa). Thus managing the various combinations of registration systems and applications requires a standard communication protocol based on registration primitives. This makes it easy to intermix the different components. In GroupKit, we define two messages or events that are sent from the registration system to the application. The first instructs the application to add a new user to the groupware session, with a special instruction to one of the replicated applications to update the new arrival. The second message instructs the application to remove a user from the session. Though the way in which these two events are generated varies with the registration module, the applications need only be concerned with handling these specific events.

Minimize Information Sent to Application. In order to provide the high interoperability between quite different registration and application systems, we have tried to keep to a minimum the amount of critical information communicated about the meeting and its participants. This greatly simplifies the construction of both applications and registration systems, at the cost of neglecting information that particular applications could want. To compensate, the registration system can augment the required information sent to the application, and applications may choose to take advantage of this information if present. However, any extra information provided by the registration system must be viewed as supplementary by the applications, and applications must be able to work with only the minimal required information. We see this as a reasonable trade-off between inter-operability and the potential for richer systems.

Support a Multiplicity of Behaviors and Interfaces Through Open Protocols. One of the key architectural features of GroupKit's registration scheme is the ability to specify a very wide variety of behaviors, policies and interfaces within a single system. A single Registrar can support a number of different Registrar Clients, each behaving quite differently. The result is that different users may have quite different registration tools, even within the same meeting. The system relies on a technique called "open protocols," which we have described in detail elsewhere [19]. Briefly, open protocols provide us with the ability to define some very sophisticated sets of behavior based on primitive manipulations of the Registrar's information. The same Registrar can provide service for a number of differently behaved Registrar Clients, and these Clients can even interact with each other through the Registrar.

Example Registration Schemes

This section illustrates several registration schemes we have designed. We are *not* recommending these examples as a list of optimal registration methods — while some are in use, most are prototypes and have not undergone any user testing. Instead, our goals are to illustrate the diversity of registration schemes that can be implemented with GroupKit, to better understand the design and implementation of groupware registration strategy, and to show by example how GroupKit can accommodate the wide range of user requirements for registration.

Open Door Registration. Our first and most commonly used scheme provides "open door" registration. All users see what conferences are in progress, where a conference consists of a number of participants and a single groupware application. Anyone can create a new conference, or join any existing conference (hence the "open door" policy).

The interface used for this scheme is shown in Figure 4. The list on the left displays all existing conferences, and selecting one of the conferences displays its users in the list on the right. Users can join an existing conference by double-clicking on its name. To create a new conference, users select "New", which brings up the dialog box shown in Figure 5. They can then choose from a list of all the available groupware applications, and give their conference a name.

This registration scheme works well for fairly informal groupware sessions among colleagues of equal status (which has been the majority of our use). While it is best for planned encounters, it does support a modest level of casual interaction — the list of conferences and attendees can be considered opportunities for contact. Of course, it is not appropriate for all cases, e.g. when access control is required for meetings discussing sensitive information. Further, the concept of a "conference" maps to a single groupware application, so if a group wants several tools they will have to create and join several conferences.

Facilitator Controlled. Our second example takes a much different approach, mimicking group support meetings

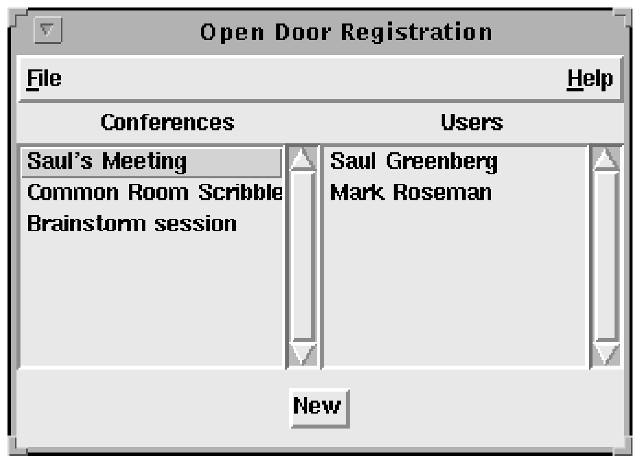

Figure 4. Open registration interface.

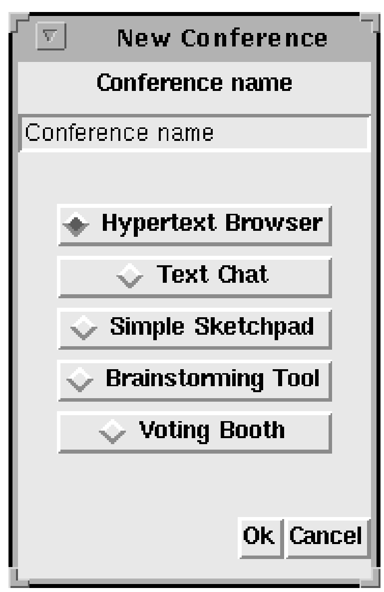

Figure 5. Dialog to start up new application.

Figure 6. Participant's Registrar Client interface to the facilitated meeting.

where a dedicated (human) facilitator manages the use of the group's tools. Under this scenario, the facilitator creates a *facilitated meeting* to which users are permitted to join — at the facilitator's discretion. The facilitator thus controls who is allowed to join the meeting, and can remove participants from the meeting at any time.

A facilitated meeting encapsulates a number of groupware applications as well as instances of two different Registrar Clients. Initially, the meeting contains no applications, and participants only see a small window indicating they are part of the meeting (Figure 6).

The facilitator sees a much different interface (Figure 7). Along the left is a list of all the users currently in the meeting; the "boot" icon allows the facilitator to remove any user (admittedly, not an icon we're likely to see in any real business meetings). The facilitator may create a number of groupware tools that can be used by the meeting participants (through the "New" menu, its contents similar to the list in Figure 5).

When a new tool is created, it appears only on the facilitator's screen, and an extra column appears on the facilitator's control panel. By toggling the checkboxes, the facilitator joins or removes individual users from tools, effectively showing or hiding the window on their screen.

With this scheme, there is strict control over who can join the meeting, making it more suitable for formal meetings than the open door scheme. Users are also removed from the burden of creating and joining conferences, relying on

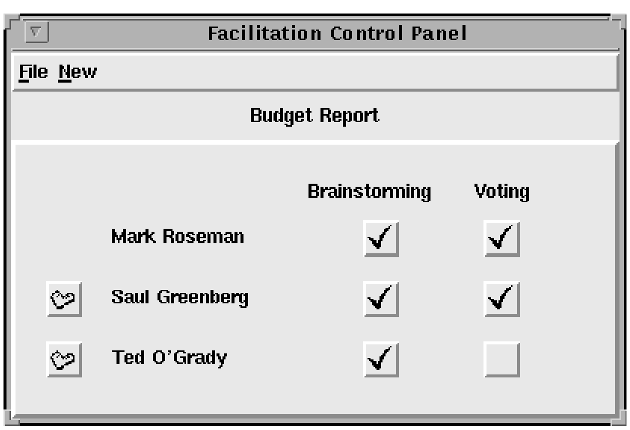

Figure 7. Facilitator's Registrar Client interface.

```
janu% cmdrc -confs
GroupSketch
Saul's Meeting
janu% cmdrc -join GroupSketch
```

Figure 8. Command line registration interface.

the facilitator to control this. Yet this scheme obviously restricts the flexibility of individual participants.

Command Line and Scripting. Our next example provides a command line interface that can be invoked from the Unix command prompt or a shell script. Presenting no graphical interface, it is a simple mechanism for creating or joining conferences with a minimum of user interaction (Figure 8).

Using command line registration, users could place a single line in the shell scripts that are executed when they login to automatically join the sketchpad session. This scheme makes it easy to join "persistent" groupware sessions. For example, consider a group that wants to keep a common blackboard (a group sketchpad) on display at all times, used perhaps to jot down notes that may trigger focused collaboration. The act of a person logging on or off (which executes the appropriate `cmdrc` command) will have them join and leave the blackboard session.

A second use of command line registration is to allow other programs to easily invoke groupware applications. Such scripting ability can be useful in a number of situations. A mail program might allow invoking a groupware application as an alternative to replying to mail, and a graphical toolbar may provide shortcuts to any number of commonly used groupware applications.

A Rooms-Based Facility. This example illustrates a spatial metaphor modeled after one of the schemes found in Cavecat [11] and other Rooms-based systems [2]. The registration interface (Figure 9) is a browser containing a number of different "rooms," which may or may not have some relation to physical rooms in a building. Each room contains any number of groupware applications. Users are free to move between rooms. When they enter a room, they are automatically joined to all applications running in the room. When they leave, they are automatically removed from these applications. As well, the group in each room can set a "privacy indicator" (indicated by how far open the door is) to tell the curious how well the group will accept newcomers. While this now capitalizes on social protocols, it could easily be extended to an actual access control mechanism.

Rooms-based registration has the advantage of grouping sets of groupware applications into work areas. It can also take advantage of spatial cues, so that several aspects of the same project might be placed in adjacent, directly linked rooms. We expect this is a useful registration

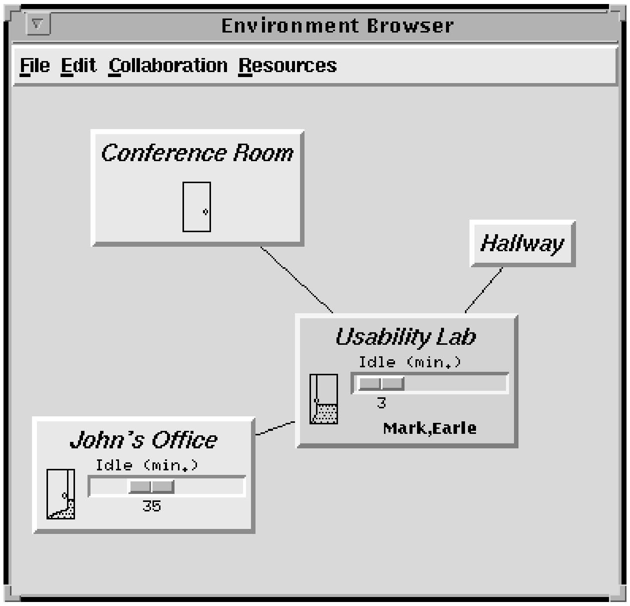

Figure 9. Prototype rooms-based registration interface.

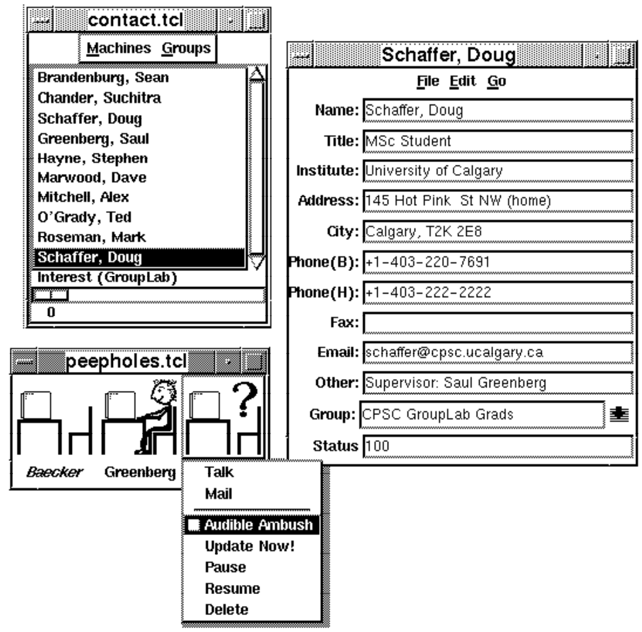

Figure 10. Contact registration interface.

system for managing a number of distinct projects or work items that persist over a long period of time, as it organizes both projects and required tools. While it may be less useful for short term groupware sessions requiring special tools (the overhead of setting up the rooms outweighs the benefits), special "breakout" rooms could be designed.

Making Contact. Our final registration example is embodied in a system called "Contact," derived from our earlier work on informal contact facilitation [3]. Contact's goal is to minimize the problem of making contact with potential collaborators, using information freely available on the Internet and low bandwidth communication channels.

Figure 10 shows the main Contact facilities. The top window shows a list of all the people the system knows about, which can be filtered based on a number of user-definable criteria to create "communities" of interest. Names are linked to a database, and contact information (addresses, phone numbers, email, etc.) can be retrieved for particular users (rightmost window).

Users can raise a "peephole" on particular users (bottom window). Peepholes provide a poor man's version of Portholes [6], a system which uses video snapshots to provide awareness of who is around and what they are doing. Peepholes provide a graphical indicator of whether the particular user is logged in and if so, iconic representations of the likelihood of them actually being by their computer (based on the elapsed time since they last touched the keyboard and mouse). This gives a reasonable indication of whether that person is available for real time collaboration. When a Peepholes partner is selected, groupware applications — including non-GroupKit tools such as mail and talk — can be started using the menu attached to each peephole.

This registration scheme works well in providing for casual contact, because it provides an overall awareness of who is around, whether they are available, and greatly simplifies setting up connections to them. Of course, Peepholes are loaded with issues of privacy violation, as it is easy to cross the line between awareness and surveillance.

FUTURE WORK

We see GroupKit's registration architecture not as a complete solution to the issues surrounding general registration, but rather as an attempt to understand and conceptualize what a solution would encompass. We envision a "groupware registration standard" whereby developers could build both registration and application components and have them seamlessly interoperate. In this section, we suggest areas we believe are in need of further investigation, discussion and refinement.

Understanding the Scope of Registration

We have exposed a number of features of registration, both human and technical. They are based upon some models of pre-meeting processes and casual interaction, and from our own experience. We would much rather be supported by a well-formulated theory of registration, if one could exist. Presumably, there are many issues that we have not considered that should be addressed by any registration standard.

Of greater import is whether we can justifiably consider registration in isolation from other meeting activities. Registration is situated within the pre-meeting process, which in turn is situated within the entire meeting process, which itself is embedded within a group's social and work context. We are now considering a complete meeting environment, where registration is more tightly integrated with other meeting activities [14].

Generalize and Refine Architecture

Any standard must deal with issues of message or protocol format. In GroupKit, we've based our communications protocols on a TCP/IP extension to the Tcl language on which GroupKit is based [15]; the protocol is in effect a set of Tcl commands. Presumably a general standard would rely on more general mechanisms.

Regardless of the underlying format used for message transmission, there are many refinements possible in the content of the messages. We have advocated keeping the amount of data passed between registration and application systems to a minimum, yet it is unclear if the set of information we have chosen is optimal. Further, some standardization of "optional" data — conventions to allow better interoperability — would be beneficial.

Tools to Support Registration Programming

Assuming for the moment a reasonable registration standard, where underlying protocols — their format and content — were well defined, there are still issues involved in building new systems. Constructing new registration components in GroupKit for example, could be much simpler than it now is. Though the interface between registration and application has been rigidly defined, encoding complex registration policies in terms of that interface (as well as the open protocols technique used within GroupKit) can be daunting, for we do not now hide much of the complexity of distributed communication.

We are now redesigning the programmer's interface to GroupKit's registration mechanism. Program libraries are being developed that encapsulate simple but powerful registration concepts. Eventually, we would like to give end users, who understand their own needs, the power to customize their own registration system.

Experiment with Registration Metaphors

Though we presented a number of different registration prototypes, none have undergone any form of user evaluation. Developing effective registration systems for particular groups will involve studying the effectiveness of the metaphors we choose, as well as the particular implementations. However, as Grudin [9] has suggested, these types of evaluations are exceedingly difficult.

CONCLUSIONS

We have tried to focus attention on the problems inherent with groupware registration, an area largely ignored by groupware researchers to date. We viewed registration as an essential component in the pre-meeting and casual interaction process, and as such it can play a large role in the success of the entire meeting. We argued that registration is highly dependent on different groups and their situations, and that an inappropriate style of registration can seriously impede or disrupt the resulting collaboration.

We derived a set of requirements for a general groupware registration system, supporting critical registration functions and the necessary diversity. We then described the registration architecture we constructed in GroupKit to address these requirements, and illustrated a number of specific registration systems we have developed.

It is our hope that this paper will start the discussion — both within the CSCW community, as well as communities concerned with the technical side of network standards — that will lead to a technical groupware registration "standard" founded on the social realities of meeting registration.

SOFTWARE AVAILABILITY

GroupKit is available via anonymous ftp from ftp.cpsc.ucalgary.ca, in the directory pub/grouplab/software.

ACKNOWLEDGMENTS

This research was supported by the National Sciences and Engineering Research Council of Canada, which funds the Grouplab and GroupKit project through its strategic and operating grant program. Special thanks to Ted O'Grady for comments on this paper as well as his work on the Groupware Environment, and Alex Mitchell for building the Rooms registration module.

REFERENCES

1. Bostrom, R., R. Anson, and V. Clawson, *Group facilitation and group support systems*. 1991, Department of Management, University of Georgia.

2. Card, S.K., M. Pavel, and J.E. Farrell, *Window-Based Computer Dialogues,* in *Readings in Human-Computer Interaction: A Multidisciplinary Approach,* R. Baecker and W.A.S. Buxton, Editor. 1987, Morgan Kaufmann.

3. Cockburn, A. and S. Greenberg. *Making Contact: Getting the Group Communicating with Groupware*. in *Conference on Organizational Computing Systems (COOCS '93)*. 1993. Milpitas, California.

4. Crowley, T., E. Baker, H. Forsdick, P. Milazzo, and R. Tomlinson. *MMConf: An infrastructure for building shared applications*. in *Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW '90)*. 1990.

5. Curtis, P. *Mudding: Social Phenomena in Text-Based Virtual Realities*. in *Proc. of the 1992 Conference on the Directions and Implications of Advanced Computing*. 1992. Berkeley, CA.

6. Dourish, P. and V. Belloti. *Awareness and Coordination in Shared Workspaces*. in *Proc. of the Conference on Computer Supported Cooperative Work (CSCW '92)*. 1992.

7. Dubs, S. and S.C. Hayne. *Distributed Facilitation: A Concept Whose Time Has Come*? in *Proc. of the Conference on Computer Supported Cooperative Work (CSCW '92)*. 1992.

8. Engelbart, D. and W.K. English. *A research center for augmenting human intellect*. in *Proceedings of the Fall Joint Computer Conference*. 1968. San Francisco, Calif.

9. Grudin, J., *Groupware and social dynamics: eight challenges for developers.* Communications of the ACM, 1994. **37**(1): p. 93-105.

10. Kraut, R.E., C. Egido, and J. Galegher, *Patterns of contact and communication in scientific research collaborations,* in *Intellectual Teamwork: Social Foundations of Cooperative Work,* J. Galegher, R.E. Kraut, andC. Egido, Editor. 1990, Lawrence Erlbaum Associates: p. 149-172.

11. Louie, G., M. Mantei, and A. Sellen, *Making Contact in a Multi-media Environment.* Behaviour and Information Technology, 1993.

12. National Centre for Supercomputing Applications, *Collage (Computer Software).* 1992.

13. Nunamaker, J.F., A.R. Dennis, J.S. Valacich, D.R. Vogel, and J.F. George, *Electronic meeting systems to support group work.* Communications of the ACM, 1991. **34**(7): p. 40-61.

14. O'Grady, T. and S. Greenberg. *A Groupware Environment for Complete Meetings*. in *Adjunct Proceedings of CHI 94*. 1994. Boston, MA.

15. Ousterhout, J.K. *Tcl: An Embeddable Command Language*. in *Proceedings of the 1990 Winter USENIX Conference*. 1990.

16. Patterson, J.F., R.D. Hill, S.L. Rohall, and W.S. Meeks. *Rendezvous: An architecture for synchronous multi-user applications*. in *Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW '90)*. 1990.

17. Root, W.R. *Design of a multi-media vehicle for social browsing*. in *Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW '88)*. 1988.

18. Roseman, M. and S. Greenberg. *GroupKit: A Groupware Toolkit for Building Real-Time Conferencing Applications*. in *Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW '92)*. 1992.

19. Roseman, M. and S. Greenberg. *Building Flexible Groupware Through Open Protocols*. in *Conference on Organizational Computing Systems (COOCS '93)*. 1993. Milpitas, California.

20. Schooler, E., *The impact of scaling on a multimedia connection architecture.* Multimedia Systems, 1993. **1**(1): p. 2-9.

21. Schooler, E. and A. Weinrib, *Multiparty Multimedia Session Control Working Group*. (Working Group Notes). 1993, Internet Engineering Task Force.

22. Suchman, L., *Plans and Situated Actions*. 1987, Cambridge University Press.