

Chapter XI

Evaluating Inter-Organizational Information Systems

Jill Drury

The MITRE Corporation, USA

Jean Scholtz

National Institute for Standards and Technology (NIST), USA

Abstract

This chapter describes different means of evaluating the usability and suitability of computer-based inter-organizational information systems (IOISs). It begins with describing why doing so is important yet difficult, and provides an assessment of the advantages and disadvantages of the major types of evaluation. It continues with a case study focusing on determining whether an application provides the necessary insight into other collaborators' identities, presence, and activities while keeping sensitive information private from a subset of the collaborators. The goal of this chapter is to provide practical guidance to organizations seeking IOISs to help them choose (or develop) an IOIS that best meets their needs.

Introduction

Information technology can be used to support collaborations and partnerships among organizations for competitive purposes. Organizations have developed the notion of inter-organizational systems (IOSs), also known as inter-organizational information systems (IOISs), to support these collaborations and partnerships. An IOIS is defined as an automated information system shared by two or more organizations (Cash & Konsynski, 1985) in a collaborative fashion.

Compare the definition of an IOIS with the definition of computer-supported collaborative work (CSCW) applications: applications that support coordinated activity carried out by groups of collaborating individuals (Greif, 1988). CSCW applications are also known as multiuser, groupware, or collaborative applications.

Collaborative applications normally provide capabilities beyond simple information access to facilitate communication and collaboration among the partners. Depending upon the collaborative application, both synchronous and asynchronous communications may be supported, and documents can be shared. Some collaborative applications incorporate video to support communications and negotiations. These coordination mechanisms are essential to efficient collaboration among cooperating organizations. In fact, because IOISs are computer-based systems used to collaborate across organizations, they are a subset of collaborative applications.

Hong and Kim (1998) built on Cash and Konsynski's (1985) work by developing a framework for classifying the various types of IOISs. Their classification scheme is based on three categories: vertical linkage, horizontal linkage, and cross linkage. Vertical systems connect suppliers with sellers with the goal of more efficient marketing. This type of system gives sellers, for example, the capability to place orders quickly and gives suppliers sales data to help them plan production. Horizontal systems link homogeneous groups of businesses. Partnerships within an industry, often consisting of smaller businesses, benefit from improved access to information. Cross systems are an attempt to integrate horizontal and vertical links into one complete system.

It is necessary to understand the roles of the participants or collaborators in IOISs in order to provide the necessary system capabilities to support a variety of tasks. For example, consider a vertical IOIS that links a manufacturer with a number of suppliers. A subset of those suppliers may be competitors who are negotiating terms with the manufacturer. Suppliers may want to use this system to share contractual information with the manufacturer but not with each other. Vertical and cross IOISs will need to support the most diverse set of users (e.g., suppliers, manufacturers, and retailers), though horizontal IOISs might also need to support differing groups of collaborators (e.g., manufacturers from the

Eastern United States versus manufacturers from the Western United States). The roles of participants and their different information sharing needs should be taken into account when evaluating which IOIS is appropriate for a set of cooperating organizations.

To help organizations evaluate which IOIS they should adopt, and provide guidance for developing an IOIS, this chapter includes an assessment of the advantages and disadvantages of using different types of evaluation methods for determining the suitability of IOIS applications. The other contributions of this chapter are as follows:

- An explanation of why IOISs are difficult, yet important, to evaluate
- A description of how the Synchronous Collaborative Awareness and Privacy (SCAPE) awareness framework could be used to evaluate an IOIS application
- A case study of evaluating the Groove™1 application's suitability for use by a collaborating team that includes members from organizations with different goals

Outline

The rest of this section discusses the importance of evaluating IOISs, the difficulties of doing so, and the critical distinctions between evaluating single-user computing applications versus multiuser applications, such as IOISs. The second section describes evaluation methods for multiuser applications in general, and one method in particular, SCAPE, will be described in the third section. The fourth section presents a case study of using SCAPE to evaluate Groove, a popular tool that aids inter-organizational information sharing. Finally, a discussion of what can be learned by evaluating IOISs completes the chapter.

The Importance of Evaluating IOISs

Much research has centered on evaluating the usability of collaborative applications, because it is extremely important to ensure that these applications can be effectively and efficiently used by their intended audiences. The success of a collaborative application normally depends on a “critical mass” of users accepting and making proper use of the application. For example, picture:

- An inventory control system so cumbersome to use that some of the staff receiving inventory neglect to log the inventory into the system or log it incorrectly
- An instant messaging application that users did not find easy and rewarding to use, so very few of a user's business or social contacts bother to remain accessible via instant messaging
- An automated calendar management application that takes a lot of work to enter activities into, so many people within a workgroup do not make an effort to keep their calendars up-to-date

Clearly, each of these situations constitutes a recipe for failure. These cases illustrate the fact that a collaborative application is likely to fail if the work people need to put into the application exceeds the perceived value of their benefits from using the application (Grudin, 1988).

Besides an imbalance in work versus benefits, there are other reasons why adoption of an IOIS may fail. For example, we have seen adoption failure of a collaborative application fail when:

- *Users did not perceive a need to collaborate:* Such a finding is consistent with Rogers' (1995) work on diffusion of innovations, which notes that the rate of adoption of innovations is related to the extent to which the innovation (e.g., a new collaborative application) satisfies users' needs.
- *The application did not provide functionality that users felt was relevant:* Relevant functionality is dependent on the tasks the users need to perform and the conditions under which they normally perform those tasks. For example, an IOIS intended for use by personnel driving delivery trucks that requires a substantial amount of typing (instead of, say, using a bar code reader) would be likely to be unsuccessful.
- *Users were not available to log in frequently:* Users who often attend meetings or engage in other activities that preclude access to the IOIS, for example, would be unlikely to embrace use of the IOIS.
- *Users did not develop a well-articulated communications strategy:* An example of an incomplete communications strategy is one that does not define situations in which to use the collaborative application versus e-mail.
- *The application was not easy to learn or use:* Rogers (1995) noted that adoption of innovations is related to the complexity or ease with which an innovation can be understood.

There is normally a large financial difference to a collaborating group of organizations between failure to adopt an IOIS and adopting—and making good use of—an IOIS that is well-suited to those organizations. When adoption failure occurs, the organizations must count as a loss the purchase price of the IOIS plus the loss in productivity represented by the hours spent installing, training, and experimenting with the IOIS. Collectively, these costs could be substantial, especially for large organizations that may have asked hundreds of people to try the IOIS. In addition, there is an intangible cost: members of the organizations may be less open to use of IOISs in the future once they have had a negative experience with an IOIS.

Contrast this failure situation with the case in which organizations choose an IOIS that meets their needs and streamlines their business processes. Depending on how an IOIS is used, an effective IOIS may result in a decreased need to travel (because IOIS technology can often mitigate the need for face-to-face meetings), shorter document production review cycles, decreased time-to-market, increased sales, and better customer support.

The difference between adoption failure and success hinges on defining collaboration requirements that take into account the work characteristics of users, the likely benefits to users, as well as ease of use from the point of view of the intended set of users. A further prerequisite for success is an evaluation program that examines how well an IOIS is likely to meet those requirements.

Evaluation goals differ depending on whether an IOIS is being chosen from among a set of existing products, or a custom (bespoke) IOIS product is being developed. If an IOIS is being chosen, the candidate applications are each examined against a tailored set of requirements, using one or more evaluation methods such as those discussed later in this chapter. Because the commercial IOISs cannot normally be modified substantially, the one that comes closest to meeting the requirements is chosen. If a custom IOIS is being developed, the goal of evaluation is to find problems that can be corrected as early as possible in the product design and development life cycle. The later in the development process that interface problems are found, the more costly they are to correct. Mantei and Teorey (1988) found that changes made to the interface designs of systems after production coding had begun cost four times as much as changes to the designs made during prototyping phases.

The Challenges of Evaluating IOIS Applications

There are several reasons why evaluating collaborative applications is more difficult than evaluating single-user computing applications. Malone (1985) cited the difficulties in assembling a group of people in a lab that reflect the social, motivational, economic, and political characteristics of typical users—yet these

characteristics are likely to affect performance when using the collaborative system. If evaluation is attempted in the users' normal work environments ("in the field"), Grudin (1988) observed that it is extremely difficult to disperse evaluators to the various locations of the collaborators as well as take into account the wide variation in user group composition and work environments. Regardless of whether they occur in a lab or in the field, Grudin (1988) noted that evaluations of collaborative applications take much more time than evaluations of single user applications, because the relevant group interactions "typically unfold over weeks."

Adding to the difficulty of collaborative application evaluation is the fact that sophisticated applications allow users to take on a number of different roles. Users' expectations of an applications' behavior may change depending upon the roles users are playing at the time and the specific tasks they are performing. More generally, collaborative applications are challenging to evaluate due to the need to take into account how the application mediates users' interactions with each other.

Although difficult to perform, evaluations of collaborative applications are extremely important due to the cost implications described above. The purpose of this chapter is to provide insight into the state-of-the-art in evaluating collaborative applications in general, and into one evaluation method in particular (which will form the basis for the case study presented later in this chapter).

The Critical Difference Between Evaluating Single-User and Multiuser (e.g., IOISs) Computer Applications

The preceding subsection touched upon the crucial distinction between evaluations of single-user systems, such as word processors, and multiuser systems, such as IOIS applications. An evaluation of multiuser (collaborative) systems needs to investigate whether the application adequately supports collaborators' awareness of each others' presence, identities, and activities. Awareness is important in collaborative applications, because it aids coordination of tasks and resources, and it assists in transitions between individual and shared activities (Dourish & Bellotti, 1992). Dourish and Bellotti (1992) defined awareness as "an understanding of the activities of others, which provides a context for your own activities." "Workspace awareness" was defined by Gutwin et al. (1995) as the up-to-the-minute knowledge of other participants' interactions with the shared workspace, such as where other participants are working, what they are doing, and what they have already done in the workspace.

To understand the importance of awareness, picture trying to use a chat application without being able to read the contributions of the other participants;

clearly, the application is useless without an understanding of the other participants' activities. Chat is an example of a synchronous collaborative application (those that are used by collaborators at the same time, although not necessarily at the same place). An "up to the moment" awareness of others' activities is especially pertinent to the class of synchronous (as opposed to asynchronous) collaborative applications. An example of an asynchronous collaborative application is e-mail.

Awareness and privacy are in tension with one another, as are awareness and information overload. Hudson and Smith (1996) expressed the trade-offs very well:

This dual tradeoff is between privacy and awareness, and between awareness and disturbance. Simply stated, the more information about oneself that leaves your work area, the more potential for awareness of you exists for your colleagues. Unfortunately, this also represents the greatest potential for intrusion on your privacy. Similarly, the more information that is received about the activities of colleagues, the more potential awareness we have of them. However, at the same time, the more information we receive, the greater the chance that the information will become a disturbance to our normal work. This dual tradeoff seems to be a fundamental one. (p. 247)

Any evaluation method that pertains to collaborative applications should be sensitive to issues of privacy and awareness. For example, a student using a distance-learning application may want to make the instructor aware of all of his or her online activities, while the instructor would want to keep grading activity private (except to the student directly affected).

It is difficult to apply evaluation techniques developed for single-user applications to multiuser applications, because they do not address the issues of awareness and privacy. Only recently has there been a push toward developing evaluation methods specifically for collaborative applications.

Usability/Suitability Evaluation Methods for IOIS Applications

An important prerequisite for applying usability and suitability evaluation methods is a thorough understanding of the users' requirements and desires. Thus, the

first part of this section provides a brief discussion of how to acquire such an understanding.

The remainder of this section provides an overview of the three broad categories of evaluation methods: formal methods (analytic methods such as dialogue and task modeling techniques), empirical methods (experiments and user studies involving human test subjects), and inspection methods (expert examination of user interfaces). Despite the difficulties enumerated above, some usability evaluation methods have been developed for collaborative systems.

Understanding Users' Needs

There are many techniques for learning about users' collaboration needs. Some of them are as follows:

- **Task analysis:** A family of different techniques that involve breaking apart users' tasks, from the standpoints of either cognitive or physical activities, at a high level of abstraction or in great detail, depending upon the particular task analysis technique chosen. For practical advice on performing task analyses, we recommend Mayhew (1999).
- **Ethnographic observation:** A broad-based approach originating in anthropology in which users are observed while they pursue their normal activities; observers become participants by immersing themselves in the users' environment. For examples of ethnographic observation applied to adoption of collaborative applications, see the work of Bonnie Nardi [e.g., Nardi and O'Day (1999)].
- **Contextual inquiry:** An ethnographic-based technique in which the observer becomes an apprentice of the person being observed; besides observation, contextual inquiry involves focused interviews, discussion, and reconstruction of past events (Holtzblatt & Jones, 1993).
- **Critical incident interviews:** A method in which users are interviewed about the events and activities surrounding an unusual or high-impact event. Klein (2000) described the use of critical incident interviews for collaborating teams.

Without understanding users' characteristics and work environments, it is impossible to determine whether an IOIS would be "natural" or "intuitive" for those users, or whether the IOIS would be compatible with the users' normal work practices. Consider an application targeted at scientists and mathematicians, such as Mathematica™. Mathematicians expect to see terminology in the

interface such as “factorial” and “cosine”; they do not need definitions of these terms. Factorial and cosine functions also exist in ExcelTM, which was designed for a general audience. In Excel, mathematical terms are defined, and the definitions are readily visible (they are not buried in a “help” file, for example).

Formal Methods

An example of a formal method that can be used to evaluate collaborative systems is Critical-Path-Method (CPM)-Goals, Operators, Methods, Selection rules (GOMS) (John & Kieras, 1994). CPM-GOMS is also known as Cognitive-Perceptual-Motor-GOMS because of its purpose to model the parallel, multi-stage processor nature of human information processing. CPM-GOMS is a task modeling technique that allows the analyst to break down a task at a very fine level of granularity, such as individual eye and hand movements. The method does not assume that each subtask happens serially; it takes into account the parallel nature of performing activities (e.g., both hands can be moving at the same time while eye movement is also occurring). The end results are predictions for task execution times. While CPM-GOMS was originally envisioned as a method for analyzing a task performed by an individual, its assumption of parallelism lends it to analyzing a task performed by a team of individuals.

An advantage of using a formal method such as CPM-GOMS is the fact that no user participation is required, which simplifies the problem of trying to recruit and schedule users and either replicate a realistic work environment in the lab or capture all facets of the users’ environment in the field. A disadvantage of using a formal method is that evaluators normally require extensive training in the method, because the methods are usually complex and can require grounding in specific theories. CPM-GOMS is useful for obtaining a detailed understanding of how quickly a particular task can be done using an interface but cannot be used to answer broader questions such as, “how satisfied will the users be with this interface?”

Empirical Methods

We are not aware of any empirical methods that have been tailored or created specifically for collaborative systems. Some researchers have applied empirical methods developed for single-user systems to small-scale collaborative systems with success. For example, Gutwin and Greenberg (1998) performed usability testing to compare two different interface approaches for a collaborative computer-assisted welding application. In general, usability tests consist of typical users performing typical tasks under controlled, but realistic, conditions,

either in a laboratory or in the field. In Gutwin and Greenberg's test, the subjects worked in pairs in two different locations and performed their tasks over the course of a few hours. The study goals were focused enough to involve only two people at a time performing a few tasks with minimal training. As a result, the study conductors could obtain a rich amount of data and insights within a manageable time period.

Usability tests are often considered to be the "gold standard" in terms of the amount of data and the subtlety of problems that may be uncovered; thus, they are advantageous to perform whenever it is not too difficult to do so. The difficulties normally arise when duplicating a realistic environment of use, recruiting appropriate users, and scheduling them in groups. The challenges only increase when "typical" user group sizes rise; testing with two to five people at a time is much more tractable than 50, for example.

Although it is highly desirable to perform realistic usability tests, such tests have proven to be too difficult and expensive to perform on collaborative applications in many cases. Baker, Greenberg, and Gutwin (2001) stated, "we have not yet developed techniques to make groupware [collaborative] evaluation cost-effective within typical software project constraints." Thus, collaborative applications are often developed or chosen without any evaluation whatsoever. Recent work in tailoring inspection methods for collaborative applications has taken place in an attempt to provide a reasonable means for collaborative application evaluation to take place.

Inspection Methods

Inspection methods are promising, because they can often be performed more quickly and inexpensively than the other usability evaluation methods. Savings accrue because they do not involve scheduling users (as do empirical methods) or extensively training evaluators (often needed for formal methods). They are often used when there is insufficient time or budget to perform usability testing or to analyze an interface using a formal method. Further, they are often used early in the development process on low-fidelity prototypes to gain early insight into whether the proposed design is consistent with general principles of human-computer interaction (even if empirical evaluations are scheduled for later versions of the application). The disadvantage with inspection methods is that they do not always result in finding the subtle problems that occur due to mismatches between the application design and the user's mental model of how the application is working.

The classic inspection method is heuristic evaluation (Molich & Nielsen, 1990). It is useful to describe it, because several methods have been developed for evaluating multiuser systems that are adaptations of heuristic evaluation. When

performing an heuristic evaluation, inspectors (often, but not necessarily, usability specialists) judge whether each user interface element conforms to established usability principles known as heuristics. Examples of heuristics are, “The interface should be consistent,” and “The interface should provide clearly marked exits.” To apply the heuristics, individual evaluators independently step through all parts of a user interface, noting cases where the interface violates the heuristics. After looking at the interface, each evaluator may assign a score to how well the interface meets each heuristic in general. Once each individual assessment is complete, evaluators normally discuss their findings and agree upon a joint set of problems and scores. The power of this method comes from combining the observations of several inspectors, because people normally find somewhat different subsets of the problems. Heuristic evaluation is straightforward enough that people other than human–computer interaction experts or human factors engineers can successfully perform an heuristic evaluation with as little as an hour’s training.

Other inspection methods compare an application against a set of guidelines (either general or application-specific) or a “capabilities” (function) checklist tailored to the users’ needs. An example of a tailorable function checklist for collaborative applications can be found in Drury et al. (Drury, Damianos, Fanderclai, Hirschmann, Kurtz, & Linton, 1999).

Three inspection methods developed for collaborative systems employ heuristics-based inspection: benchmarks for workspace awareness (Villegas & Williams, 1997), the Locales Framework heuristics (Greenberg, Fitzpatrick, Gutwin, & Kaplan, 2000), and the “Mechanics of Collaboration” (Baker, Greenberg, & Gutwin, 2001). An additional inspection method, Synchronous Collaborative Awareness and Privacy Evaluation (SCAPE) (Drury, 2001) provides both a means of specifying awareness and privacy requirements and evaluating whether the application satisfies the requirements via an heuristic approach. We describe two methods in more detail, the Mechanics of Collaboration and SCAPE, because they are more recent and more mature than the others.

Gutwin and Greenberg (2000) maintained that there are some basic collaboration activities that should be supported by any collaborative application:

These activities, which we call the mechanics of collaboration, are the small scale actions and interactions that group members must carry out in order to get a shared task done. Examples include communicating information, coordinating manipulations, or monitoring one another. (p. 98)

Gutwin and Greenberg proposed that the mechanics of collaboration framework can be used to construct heuristics. They formed eight heuristics (Baker, Greenberg, & Gutwin, 2001, p. 125):

- Provide the means for intentional and appropriate verbal communication
- Provide the means for intentional and appropriate gestural communication
- Provide consequential communication of an individual's embodiment
- Provide consequential communication of shared artifacts
- Provide protection
- Provide management of tightly and loosely coupled collaboration
- Allow people to coordinate their actions
- Facilitate finding collaborators and establishing contact

The idea behind the Mechanics of Collaboration method is that evaluators inspect the interface using the heuristics from Baker, Greenberg, and Gutwin (2001) instead of the ones developed by Molich and Nielsen (1990) or, more recently, Nielsen (1994). Otherwise, the method is essentially the same as that developed by Molich and Nielsen (1990).

Note that the heuristics of Baker, Greenberg, and Gutwin (2001) are broad and make the assumption that the role of the user is not a factor in the evaluation. The SCAPE method was developed to provide a finer-grained evaluation technique, acknowledging that users of an application may have different awareness and privacy needs.

The SCAPE Method

Later in this chapter, we will give an example of how SCAPE was used to evaluate the awareness and privacy support of a particular application being considered for use by a retailer and supplier. Because we will give examples of developing SCAPE analysis materials that were culled from our case study, we introduce the case study scenario in the next subsection. The fine-grained awareness framework and awareness relationships that underpin SCAPE are introduced in the following subsections, ending this section with a by a step-by-step description of the SCAPE method.

Case Study Scenario

We developed a scenario based on recent events in the popular business press (e.g., (Bianco & Zellner, 2003)). The scenario centers on the partnership between a fictitious major retailer (“Wal-Store”) and an equally fictitious nationwide distributor of juice drinks (“Sea Spray”).

Joy Brown is a beverages buyer at Wal-Store, and Jenn Smith is a manufacturer's representative from Sea Spray. Jenn works with two product managers, Rod Leeds and Sally Steele. Joy wants to get the Sea Spray juice drink products on the shelves at Wal-Store at the lowest possible price by probing Sea Spray's manufacturing, inventorying, and distribution processes and suggesting ways to streamline these processes. Jenn would like to get as many different Sea Spray products on Wal-Store's shelves as possible and so is willing to work with Joy—up to a point. She does not want to give Joy information about the recipes for the newest juice drinks because Wal-Store has a history of using “inside” product information to manufacture similar products under the “Sal's Choice” in-house label, thus eating into the market share of national brands.

Jenn would like to use the same IOIS with Joy as well as Rod and Sally, her product managers for the two new juice drinks. She would like to keep the proprietary information private from Joy, however.

By highlighting the natural tension between competitors, this Wal-Store scenario includes the notion of *adversarial collaboration* (Cohen, Cash, & Muller, 2000): situations in which collaborators have widely divergent goals yet must work together to perform specific tasks. Other situations that involve adversarial collaboration are merger/acquisition negotiations and document sharing among opposing legal teams. We expect that, as technology becomes more widely used to facilitate communications between organizations with divergent goals, computer-based adversarial collaboration support will become increasingly more important.

Both Cohen et al.'s study and the Wal-Store scenario illustrate a need for IOISs to support detailed awareness and privacy requirements and to ensure that everything that is visible to others is revealed intentionally instead of being revealed accidentally or by default.

Awareness Framework

Before describing the SCAPE method, we first need to refine the definitions of awareness cited earlier into a more fine-grained analysis framework. Although there are many definitions of awareness (a few of which were cited above), there is no standard definition. We define awareness as follows (Drury, 2001):

Awareness - Given two participants p_1 and p_2 who are collaborating via a synchronous collaborative application, awareness is the understanding that p_1 has of the identity and activities of p_2 .

To support p_1 's understanding of p_2 , an application provides p_1 with information about the identity and activities of p_2 *without* p_1 *having to request the information or* p_2 *having to explicitly transmit it*. Awareness information is intended to emulate the kinds of nonverbal cues that people get about each other when they work face to face in the same physical environment.

The awareness information that p_1 might have access to regarding p_2 includes (but is not necessarily limited to) p_2 's presence in the shared workspace, p_2 's identity, the task that p_2 is performing, the tools and artifacts that p_2 is using, the changes p_2 is making, the area of the workspace viewable by p_2 , and p_2 's focus within that viewable area. We refer to the aggregation of all of the awareness information that all of the participants may have about each other as the *awareness information space*.

Although the awareness literature largely assumes that the more awareness information available, the better, there are times when awareness information should be withheld. For our purposes, we define *privacy* as follows:

Privacy - The intentional withholding of awareness information.

Evaluation of awareness support has two underlying principles: (1) ensure that awareness information that should be provided is provided; and (2) ensure that awareness information that should not be provided is kept private. A different type of awareness-related error is associated with each of these principles (Drury & Williams, 2002):

Type 1 - Awareness violation. Awareness information that should be provided is not (a violation of the first principle).

Type 2 - Privacy violation. Awareness information that should be kept private is revealed (a violation of the second principle).

A method for evaluating awareness support must provide for the identification of awareness violations and privacy violations.

There is currently no theory *per se* of awareness support. We constructed a general-case default specification for the kinds of awareness information after examining eight theories and frameworks for collaborative work [see Drury & Williams (2002)]. We developed a general, application-independent awareness specification, expressed as heuristics (see Figure 1). These *awareness heuristics* are of two types: *activity heuristics* pertaining to activities only, regardless of who is performing them; and *identity heuristics* pertaining to participants.

Figure 1: General awareness specification, expressed as activity heuristics and identity heuristics. This specification is application-independent.

Type	Heuristic
Activity	Show the tasks being performed
	Show the tools being used
	Show the changes being made
	Show the historical changes made
	Show the time of each historical change
Identity	Show participants' identities
	Show the immediate intentions of each participant
	Show the focus of each participant
	Show area viewable by each participant
	Show the participant(s) performing each task
	Show the participant(s) using each tool
	Show the participant(s) making changes
	Show the participant(s) who made each historical change

The general awareness specification has two assumptions built into it: (1) all participants using a collaborative application have the same awareness needs, and (2) participants require access to all possible awareness information about each other. The general specification may be tailored for specific applications where these assumptions do not hold.

We refer to the situation where a participant has access to all possible awareness about another participant as *complete awareness*:

Complete awareness - Given two participants p_1 and p_2 who are collaborating via a synchronous collaborative application, if all information regarding p_2 in the awareness information space is available to p_1 , we say that p_1 has complete awareness with respect to p_2 .

It is common for a participant to need (or to be permitted) only limited, rather than complete information about another participant, so that *partial awareness* should be provided:

Partial awareness - If some, but not all, information regarding p_2 in the awareness information space is available to p_1 , we say that p_1 has partial awareness with respect to p_2 .

Whether a participant p_1 needs complete or partial awareness of another participant p_2 depends on the *roles* played by p_1 and p_2 . For example, in the application supporting the retail scenario, p_1 may be a buyer and p_2 may be a sales representative. We define a *role* to be a participant's activities and responsibilities with respect to the other participants in a collaborative session.

Participants using a collaborative application can be partitioned into role-based equivalence classes (e.g., buyer, sales representative). All participants in the same role have the same awareness and privacy needs.

Awareness Relationships

Each role is related to every other role by an *awareness relationship* that characterizes the awareness information participants in one role may have about participants in another.

An awareness relationship may be either complete or partial, depending whether participants have complete or partial awareness of each other. It is also possible for participants in one role to have no awareness information about participants in another role, in which case we describe the awareness relationship as *no awareness*.

Awareness relationships are unidirectional. Given two roles r_1 and r_2 , there is an awareness relationship for r_1 with respect to r_2 , characterizing the information available to participants in r_1 about participants in r_2 . The relationship may be complete, partial, or no awareness. Similarly, there is an awareness relationship

Figure 2: Awareness relationship matrix for the hypothetical application used by retailers and suppliers. (Numbers indicate the number of participants in each role.)

What these roles... ↓	...can know about the roles below		
	Buyer	Sales representative	Product manager
Buyer (1)	—	Partial	No awareness
Sales representative (1)	Partial	—	Partial
Product manager (2)	No awareness	Partial	Complete

for r_2 with respect to r_1 . It also may be complete, partial, or no awareness, independent of the type of the relationship for r_1 with respect to r_2 .

Figure 2 shows a matrix of awareness relationships for evaluating a hypothetical collaboration application used by buyers and sales representatives, assuming use by one participant in the buyer role, one in the sales representative role, and two in the product managers' roles. Some roles (e.g., product managers) have awareness relationships with themselves, which characterize what participants in the same role may know about each other (what one product manager may know about another). Because there is only one buyer (and one sales representative), there is no buyer-buyer (or sales representative-sales representative) awareness relationship.

Steps for Performing a SCAPE Evaluation

SCAPE is based on the awareness framework and awareness relationships. It can be used to determine whether users' awareness and privacy needs would be met by an application. In particular, SCAPE is especially useful for evaluating a synchronous collaborative application, because such systems require an up-to-the-moment awareness of fellow participants' identities and activities. Figure 3

Figure 3: Awareness needs in synchronous versus asynchronous applications. ("Yes" means the heuristic is applicable; "No" means the heuristic is not applicable.)

Heuristic	Sync.	Async.
Show the tasks being performed	Yes	No
Show the tools being used	Yes	No
Show the changes being made	Yes	No
Show the historical changes made	Yes	Yes
Show the time of each historical change	Yes	Yes
Show participants' identities	Yes	Yes
Show the immediate intentions of each participant	Yes	No
Show the focus of each participant	Yes	No
Show area viewable by each participant	Yes	No
Show the participant(s) performing each task	Yes	No
Show the participant(s) using each tool	Yes	No
Show the participant(s) making changes	Yes	No
Show the participant(s) who made each historical change	Yes	Yes

illustrates the difference in awareness needs for synchronous versus asynchronous applications. A “yes” under the “Sync.” column means the heuristic is applicable to synchronous applications; similarly, a “yes” in the “Async.” column implies that the heuristic pertains to an asynchronous application. A “no” in either column means the heuristic is not relevant for that type of collaborative application.

SCAPE is an inspection method performed by evaluators, and it is based on the awareness framework described earlier in this section. It is designed to help evaluators find both awareness violations and privacy violations. SCAPE takes into account the roles that participants play and the awareness relationships between the roles.

The SCAPE method has two parts: (1) development of a detailed, application-specific specification of awareness and privacy requirements; and (2) evaluation of the application for compliance with the specification. The SCAPE Handbook (Drury, 2001) contains more detailed explanations than can be included here, as well as examples, advice, and worksheets.

There are three steps to developing an awareness specification for a collaborative application: define the awareness relationships, develop role-based awareness policies, and identify activity-based exceptions to the policies.

Step 1 - Define awareness relationships. The goal of this step is to identify roles and the high-level awareness relationships between them. Knowledge of the application domain is needed to perform this step. Roles are identified, and an awareness relationship matrix, such as the one shown in Figure 2, is created. The output of this step is the matrix.

Step 2 - Develop role-based awareness policies. The goal of this step is to develop a set of awareness policies, based on the relationships between roles. The approach is to modify the general awareness specification from Figure 1 according to the awareness relationships identified in Step 1. While awareness policies can be created for each role pair, in practice, usually only one awareness policy is needed; it indicates the superset of what participants can know about other participants. Exceptions to this policy are identified in the next step. The awareness policy for the case study is shown in Figure 4.

For each awareness relationship, the evaluator begins with the general awareness specification and deletes any portions of the specification that are not applicable. (For example, if the application requires anonymity, then all identity heuristics are deleted.)

Figure 4: Awareness policy for the case study. (None of the awareness heuristics are deleted in this example, and no extra heuristics have been added. Notations to the worksheet are indicated by this Comic Sans MS font.)

**Awareness Policy:
What Participants CAN Know About Other Participants
Worksheet for Step 2**

Date: 13 May 2003
Evaluator: J. L. Drury
Application: Groove

Instructions:

1. Add missing heuristics.
2. Cross out heuristics that are not applicable to participants in any role.

Identity awareness heuristics:
Show the identity of each participant.
Show the immediate intentions of each participant.
Show area viewable by each participant.
Show the focus of each participant.
Show the participant(s) performing each task.
Show the participant(s) using each tool.
Show the participant(s) making changes.
Show the participant(s) who made each historical change.

Other heuristics for identity awareness information:
None.

Activity awareness heuristics:
Show the task(s) being performed.
Show the tool(s) being used.
Show the change(s) being made.
Show the historical change(s) made.
Show the time of each historical change.

Other heuristics for activity awareness information:
None.

We also leave open the possibility that the evaluator may need to add to the specification. The evaluator performs this step as many times as necessary:

- Once for all of the complete awareness relationships, because they all have the same requirements
- Once for each partial awareness relationship, because they may all have different requirements

The output of this step is a set of role-based awareness policies expressed as heuristics.

Step 3 - Identify activity-based exceptions. The goal of this step is to tailor the role-based awareness policies developed in Step 2 so that they include any activity-related exceptions that are necessary to ensure privacy. All of the policies for partial awareness relationships must be tailored in this way. (There is no need to tailor the policies for complete and “no awareness” relationships.)

The evaluator begins with the awareness relationship matrix from Step 1 and the set of awareness policies from Step 2, and uses domain knowledge about tasks that participants will perform. The evaluator identifies the activities during which parts of an awareness policy should be suspended. (For example, it may be OK for a product manager to be aware of what a buyer is doing, *except* when the buyer is working on another purchase in which the product manager is not involved.)

Figure 5: Sample privacy policy for the case study, showing what the buyer cannot know about the sales representative's activities.

**Privacy Policy: What Participants
CANNOT Know About Other Participants
Worksheet for Step 3**

Date: 13 May 2003

Evaluator: J. L. Drury

Application: Groove

Role pairs analyzed: What Buyer CANNOT know about Sales rep.

Instructions:

- List privacy needs as exceptions to applying the heuristics.
- Note impacts to violating the privacy needs.

Privacy Needs Table	
Apply the <i>identity</i> awareness heuristics except for information related to the following activities:	Impact of privacy violation (low, medium, or high)
Sales rep's bottom-line price (unless he or she chooses to reveal it deliberately)	High
Manufacturer's recipes for juice drinks	High
Apply the <i>activity</i> awareness heuristics except for information related to the following activities:	Impact of privacy violation (low, medium, or high)
Same as above	

The result can be seen in Figure 5, which is a portion of a privacy policy for the case study. The set of tailored awareness and privacy policies constitutes an application-specific awareness requirements specification.

There are three steps to evaluating an application for compliance with an awareness specification: assess the level of effort a complete evaluation would entail, develop scenarios to use during evaluation, and perform the evaluation.

Step 4 - Assess level of effort. The goal of this step is to assess the level of effort that a full-fledged evaluation would entail. A SCAPE evaluation involves a series of mini-evaluations, because each awareness relationship needs to be evaluated. The level of effort can be substantial, because the worst-case number of awareness relationships can be roughly estimated as the square of the number of roles. Thus, it is prudent to calculate the level of effort needed for a complete SCAPE evaluation, compare the result to the resources available, and select the highest-priority awareness relationships to evaluate.

The evaluator begins with the role relationship matrix from Step 1, the specification from Step 3, and an understanding of resources (time, money, etc.) available. If the evaluator expects that evaluating one awareness relationship will take e effort (measured in evaluators' time, money, etc.), and if there are a awareness relationships, then the level of effort for a complete evaluation is proportional to $e * a$.

To identify the awareness relationships that are the highest priorities for evaluation, the evaluator annotates the awareness policies, indicating whether violations would have high, medium, or low impact. The output is a list of prioritized awareness relationships.

Step 5 - Develop scenarios. The goal of Step 5 is to specify activities that will cause SCAPE evaluators to perform sequences of actions that will exercise the application's awareness support capabilities. Scenarios are a well-established technique for evaluation; Nielsen pointed out the advantages of using scenarios for heuristic evaluation when it is important to examine participants' interactions (Nielsen, 1995). The evaluator uses the awareness specification from Step 3 and the prioritized list of awareness relationships from Step 4 to develop a master scenario and scenario worksheets to use during the evaluation. The master scenario worksheet from the case study can be seen as an example in Figure 6.

Step 6 - Perform the evaluation. The goal of this step is to identify awareness violations and privacy violations. Teams of evaluators perform the scenarios developed in Step 5 and examine the application for compliance with the

specification from Steps 2 and 3. The output of this step is a set of problem reports.

Before performing a SCAPE evaluation, it is critical that the evaluators understand the users' tasks and roles and the applications' functionality. Without a good understanding of the users, the evaluation results may not predict how well the application will meet users' needs (that is, the evaluation may point out issues that the users would not consider problems or may miss problems entirely). To understand the users, we performed a task analysis and critical incident analysis based on structured interviews with a buyer for a large Eastern-U.S. retailer. One of the evaluators already had significant prior experience with using the collaborative application and helped train the other (supplementing training provided by the developer's tutorial information).

Figure 6: Example portion of master scenario worksheet developed for the case study

Master Scenario Worksheet for Step 5

Date: 26 May 2003
Evaluator(s): Jill Drury, Jean Scholtz
Application: Groove

Instructions:

1. Determine who will participate in what roles. Not everyone who participates needs to be an evaluator.
2. List sequences of actions or subtasks that cover all evaluated role relationships and include both typical situations and situations where violating the awareness and privacy policy would have a high impact.

Participants

Name	Role	Evaluator?
Rod	Product Manager (PM)	No
Jill	Buyer (B)	Yes
Jean	Sales Rep (SR)	Yes

Master Scenario

Time	Activity	Roles	Verify privacy/awareness needs
8:00	Discuss the new juice flavors	PM, B, SR	B cannot see juice recipes
8:20	Share manufacturing process info	PM, B, SR	B cannot see juice recipes
8:40	Discuss pricing strategies	B, SR	B cannot see SR's bottom price; SR cannot see B's top price

An explanation of the evaluation performed for the case study can be found below, after a description of the application evaluated.

Case Study: Evaluating Groove Using Scape

We evaluated Groove™ to understand how well it can support the awareness and privacy needs of a retail buyer/supplier team. Groove was chosen because there is currently a lot of interest in using Groove (Groove's customer list, as published on its Web site, shows approximately 100 organizations, including large and small businesses, nonprofits, educational, and government entities). Also, Groove's functionality is similar to that of several others in its class; it is a general-purpose collaboration application intended to be used by a wide variety of organizations.

After the Groove overview, we discuss how we performed the case study and summarize its results.

Overview of Groove Functionality

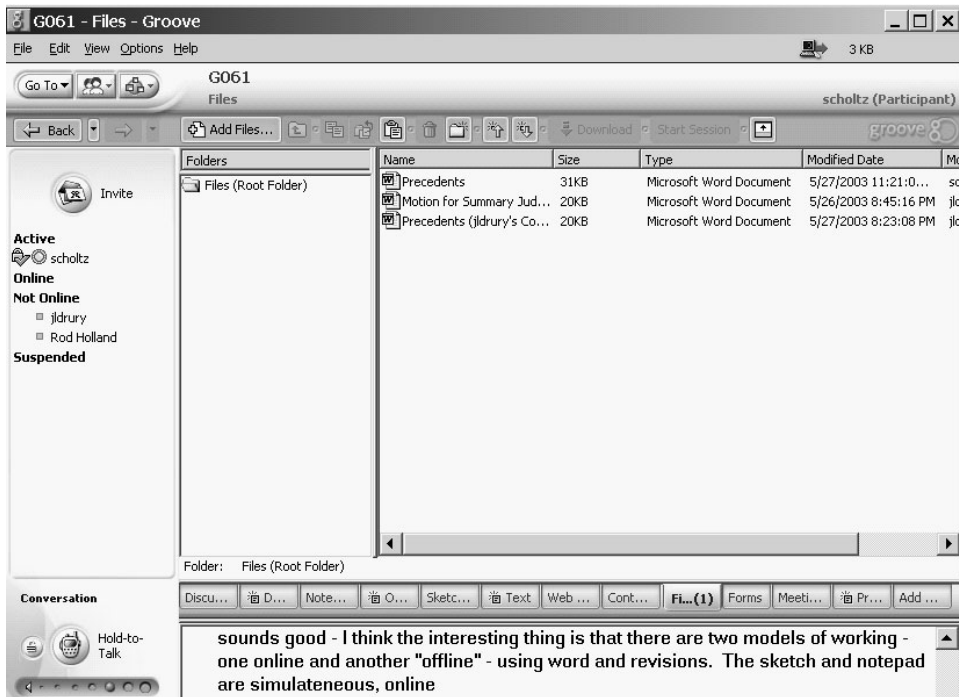
Groove peer-to-peer collaboration software provides users with synchronous as well as asynchronous collaboration. Groove uses the metaphor of shared spaces to create collaboration environments. Templates are provided for spaces, and a number of different functionalities can be incorporated into a space. Functions provided by Groove include file storage, calendar facilities, discussion spaces, sketch pads, project planning, and meeting spaces with support for agendas, minutes, and action items. A built-in browser lets participants pull up Web pages within Groove and browse the pages together.

Many of the tools available in Groove are compatible for use with Microsoft Office; these functions provide asynchronous collaboration. In addition, Groove provides text chat and audio chat for synchronous collaboration. Microsoft NetMeeting can be launched from within Groove as well.

Figure 7 shows a Groove shared space with the files tool open. The participant bar is to the left, and the audio chat tool is at the bottom left. A portion of a text chat message is shown at the bottom of the window.

Groove provides role-based access for three predefined, standardized roles. A manager sets up the shared space and can invite people as participants or guests. The privileges that one has in a shared space are based upon these roles. In general, guests are allowed read-only privileges, while participants are able to post and edit as well as read.

Figure 7: A shared Groove space



Groove has two models of working. In tools such as the sketchpad, multiple users can work together at the same time and they see each other's contributions in real time (as they are made). However, in tools such as the file space, participants open a file that creates a local copy on the user's machine. Groove contains synchronization detection that updates the document when a user stores a modified file back to the file manager.

Document creation and revision is supported via a document revision tool and an outliner. Document revision automatically creates a folder for the participants who have edited a document, separate from the folder for the original document. Documents in Groove are marked as to whether they have been read or not; this feature enables users to keep track of work they have done.

Groove also provides a "navigate together" feature. All participants who selected the "navigate together" feature will be taken from one tool in the space to another when one of the participants changes location within the space. This feature can be used during a meeting to make sure that everyone is working in the same location.

Groove provides some mechanisms for being aware of the presence, identities, and activities of fellow Groove space users. For example, there is a persistent part of the display showing the participants who are logged in, those who are

active, and those who belong to the space but are currently logged off or have been inactive for some time (suspended). The toolbar shows the number of participants viewing any particular tool space. A notice is shown when someone enters a tool space. When users are engaged in text chat, Groove shows when one of the participants is typing.

Evaluating Groove Using the SCAPE Method

Prior to evaluation, we became familiar with Groove functionality and the users tasks/roles and prepared the SCAPE materials described earlier in this chapter. We (the two evaluators) decided upon the roles we would play and recruited a third person to assist us. While not an evaluator, this third person agreed to play a role and answer specific questions relating to the levels of privacy and awareness that he observed in the Groove shared spaces during the session.

During the evaluation session, the three of us logged into a shared Groove space from three different geographic locations. We assumed the roles of a buyer, sales representative, and product manager. The buyer created the space and thus was considered the Groove Manager. The other two people were given Groove Participant privileges at first. Over the course of the session, the manager changed their Groove roles to Guest and back to Participant to see how these changes affected their awareness and privacy.

We followed the Master Scenario excerpted in Figure 6, beginning with a discussion of the new juice flavors. After performing the actions indicated in the Master Scenario worksheet, we checked to see if the privacy and awareness requirements highlighted in the worksheet were supported by Groove. We made note of the situations and conditions under which the requirements were not supported.

We communicated with each other nearly continuously during the evaluation session using the Groove chat tool. This tool provides a persistent transcript of all messages; we noted our evaluation observations in chat messages to each other and saved the chat file for post-session analysis. The evaluation took approximately 4 hours.

Results of Evaluating Groove

We combed through the chat file notes and compared our observations to the Awareness and Privacy Policies and the specific requirements noted in the Master Scenario worksheet. Highlights of the results are presented below.

Groove does a reasonably good job of providing awareness but does not support many of the privacy needs for the case in which proprietary information needs

to be seen by some collaborators but not by others. For example, Groove provides at least read-only privileges to all users for all documents. It is impossible to mark documents as being readable by only a subset of Groove users. This means that the Sea Spray team members cannot work on a document in a shared Groove space and keep that document private from members of Wal-Store's team. This situation violates the privacy need specified as, "it is OK to be able to read some documents but not others."

Of course, the problem of keeping some documents private from some collaborators can be resolved by creating new Groove "spaces" for subgroups of collaborators, such as the Sea Spray team members. This approach has the drawback of requiring collaborators to remember what documents and other information were stored in what spaces, and manage version control of documents that are stored in both spaces for the convenience of avoiding moving back and forth between spaces.

Similar to the document privacy issue, chat messages in Groove cannot be addressed to a subset of the shared space users; the messages are displayed to all users. Thus, team members from the manufacturer cannot caucus privately among themselves using the shared chat tool. This means that the privacy need specified as "The buyer cannot see the sales representative's bottom price; the sales representative cannot see the buyer's top price" cannot be satisfied if the relevant pricing material is included in the Groove space.

Groove does not provide the means for two people to have management privileges in a shared space; thus, either the buyer or sales representative would be the manager, and the other could be a participant, at best. While not violating an awareness or privacy requirement, such a situation would not be palatable to two people who are trying to achieve a status in which they are equals in their collaborations.

By default, Groove participants are allowed to invite others into the shared space. Unless the manager revokes this "invitation" privilege for everyone, it is possible for a participant to invite another to be a participant, who invites someone else, etc. Thus, it may not be known to everyone who has been invited and may subsequently join a session. Once someone has joined, their user name will be visible to everyone in the space unless that person restricts what group can see their online identity; but by that time, the damage may have been done if they were not supposed to have had any access to the materials in the shared Groove space. Imagine the consequences of the relatively uncontrolled invitation mechanism for a sensitive shared space containing pricing or budget documents.

While Groove provides awareness of who has revised documents, it does not provide notification of who has revised other shared artifacts. For example, a meeting agenda can be erased or revised by anyone—even a guest—and there is no indication that such a revision has taken place. In fact, most people would

be likely to assume that the agenda was authored by the person who scheduled the meeting; this may not be true. Also, the original author does not have any notification that their work was changed. This situation violates the heuristics in the awareness policy: “Show the participant(s) making changes” and “Show the participant(s) who made each historical change.”

Discussion

Based on our knowledge of other collaborative applications, we do not believe that Groove is unique in its lack of privacy support. Privacy is enhanced in applications that provide access control on a document-by-document basis and that provide the capability to address chat messages to subsets of participants (e.g., a private, “whispered” chat message). In general, however, most collaborative applications are akin to Groove in that they aim to provide support to participants who wish to share everything, all the time. Such an approach is not consistent with the move toward greater computer-aided collaboration across organizations with somewhat divergent goals.

Of all the evaluation methods discussed, SCAPE is the only method to specifically focus on whether an application supports the awareness and privacy needs of a particular group of users. SCAPE does not address basic usability issues such as whether the interface was designed to be consistent or to have clearly marked exits. Inspection techniques developed for single-user applications can provide this type of insight, even for multiuser systems, and should generally be used. Thus, SCAPE should be used in conjunction with techniques such as Nielsen’s heuristic evaluation and function checklists. If an IOIS will be used in situations where awareness and privacy are important concerns, however, a SCAPE analysis would be appropriate and beneficial.

SCAPE does not involve users as evaluation subjects; this is both a strength and a weakness. Usability testing (structured testing with users) can be very expensive and time-consuming but yields rich data. Rather than forgoing evaluation completely, however, an inspection evaluation method constitutes a less expensive and less time-consuming approach. We recommend that usability testing be conducted if doing so involves small numbers of users, the users’ environment can be easily recreated or accessed, and the user population is readily available. Even if usability testing is performed and a full SCAPE evaluation or other inspection evaluation is not attempted, we recommend performing the first three steps of SCAPE to understand the roles that users play and the need to provide information to, or conceal information from, each other. Once SCAPE Awareness and Privacy Policies have been developed, they can

act as a specification against which the application's performance can be compared during a usability test. Awareness and Privacy Policies can also provide guidance in determining which tasks the users should be asked to perform to ensure that sensitive or critical information is revealed or concealed in certain situations.

If the buyer and sales representative in the case study scenario were truly looking for an IOIS to manage information sharing in an adversarial collaboration environment, they would be well advised to perform SCAPE evaluations on several systems as part of a comparative analysis. At a minimum, the most sensitive situations should be included in scenarios developed for the SCAPE evaluation. For example, SCAPE quickly highlighted the fact that Groove does not provide for documents to be read by only a subset of the application's users; this fact alone means that Groove may not be a suitable choice for this particular group of users.

References

- Baker, K., Greenberg, S., & Gutwin, C. (2001). Heuristic evaluation of groupware based on the mechanics of collaboration. In M. R. Little, & L. Nigay (Eds.), *Engineering for human-computer interaction* (8th IFIP International Conference, EHCI 2001, Toronto, Canada, May 2001), Lecture Notes in Computer Science, Vol. 2254 (pp. 123–139). Heidelberg: Springer-Verlag.
- Bianco, A., & Zellner, W. (2003). Is Wal-Mart too powerful? *Business Week*, 6 October 2003, 100–110.
- Cash, J. I., Jr., & Konsynski, B. R. (1985). IS redraws competitive boundaries. *Harvard Business Review*, 63(2), 134–142.
- Cohen, A. L., Cash, D., & Muller, M. J. (2000). Designing to support adversarial collaboration. *ACM: Proceedings of the Computer Supported Cooperative Work (CSCW) 2000 conference* (pp. 31–39), Philadelphia, PA.
- Dourish, P., & Bellotti, V. (1992). Awareness and coordination in shared workspaces. *ACM: Proceedings of the Computer Supported Cooperative Work (CSCW) '92 conference* (pp. 107–114), Toronto, Canada.
- Drury, J., Damianos, L., Fanderclai, T., Hirschmann, L., Kurtz, J., & Linton, F. (1999). *Methodology for evaluation of collaborative systems, v. 4.0*. DARPA Intelligent Collaboration and Visualization Project: published at <http://zing.ncsl.nist.gov/nist-icv/documents/methodv4.htm>

- Drury, J. (2001). Extending usability inspection evaluation techniques for synchronous collaborative computing applications. Sc.D. thesis, Department of Computer Science, University of Massachusetts Lowell.
- Drury, J., & Williams, M. G. (2002). A framework for role-based specification and evaluation of awareness support in synchronous collaborative applications. *IEEE: Workshops on Enabling Technologies, Infrastructure for Collaborative Enterprises (WETICE)* (pp. 12–17), Pittsburgh, PA.
- Greenberg, S., Fitzpatrick, G., Gutwin, C., & Kaplan, S. (2000). Adapting the Locales Framework for heuristic evaluation of groupware. *Australian Journal of Information Systems (AJIS)*, 7(2), 102–108.
- Greif, I. (1988). *Computer-supported cooperative work: A book of readings*. San Mateo, CA: Morgan Kaufmann.
- Groove™ (2003). Groove Networks: www.groove.net.
- Grudin, J. (1988). Why CSCW applications fail. *ACM: Proceedings of the Computer Supported Cooperative Work (CSCW) '88 conference* (pp. 85–93), Portland, OR.
- Gutwin, C., & Greenberg, S. (1998). Effects of awareness support on groupware usability. *ACM: Proceedings of the CHI 98 Conference on Human Factors in Computing Systems* (pp. 511–518), Los Angeles, CA.
- Gutwin, C., & Greenberg, S. (2000). The mechanics of collaboration: Developing low cost usability evaluation methods for shared workspaces. *IEEE: WETICE Workshop on Collaborative Enterprises* (pp. 98–103), Gaithersburg, MD.
- Gutwin, C., Stark, G., & Greenberg, S. (1995). Support for workspace awareness in educational groupware. *Proceedings of the First CSCL Conference on Computer Supported Collaborative Learning* (pp. 147–156), Bloomington, Indiana. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Holtzblatt, K., & Jones, S. (1993). Contextual inquiry: A participatory technique for systems design. In D. Schuler, & A. Namioka (Eds.), *Participatory design: principles and practice* (pp. 177–210). Hillsdale, NJ: Lawrence Erlbaum Associates.
- Hong, I. B., & Kim, C. (1998). Toward a new framework for interorganizational systems: A network configuration perspective. *IEEE: Proceedings of the Thirty-First Hawaii International Conference on Systems Science*, Vol. 4 (pp. 92–101).
- Hudson, S., & Smith, I. (1996). Techniques for addressing fundamental privacy and disruption tradeoffs in awareness support systems. *ACM: Proceedings of the CSCW 96 Conference on Computer Supported Cooperative Work* (pp. 248–257).

- John, B. E., & Kieras, D. E. (1994). The GOMS Family of analysis techniques: Tools for design and evaluation. Pittsburgh, PA: Carnegie Mellon University.
- Klein, G. (2000). Cognitive task analysis of teams. In J. M. Schraagen, S. F. Chipman, & V. L. Shalin (Eds.), *Cognitive task analysis* (pp. 417–429). Mahwah, NJ: Lawrence Erlbaum Associates.
- Malone, T. W. (1985). Designing organizational interfaces. *ACM: Proceedings of the CHI 85 Conference on Human Factors in Computing Systems* (pp. 66–71), San Francisco, CA.
- Mantei, M. M., & Teorey, T. J. (1988). Cost/benefit analysis for incorporating human factors in the software lifecycle. *Communications of the ACM*, 31(4), 428–439.
- Mayhew, D. (1999). *The usability engineering lifecycle: A practitioner's handbook for user interface design*. San Francisco, CA: Morgan Kaufmann.
- Molich, R., & Nielsen, J. (1990). Improving a human–computer dialog. *Communications of the ACM*, 33(3), 338–348.
- Nardi, B. A., & O'Day, V. L. (1999). *Information ecologies: Using technology with heart*. Cambridge, MA: MIT Press.
- Nielsen, J. (1994). Enhancing the explanatory power of usability heuristics. *ACM: Proceedings of the CHI 94 Conference on Human Factors in Computing Systems* (pp. 152–158), Boston, MA.
- Nielsen, J. (1995). Scenarios in discount usability. In J. M. Carroll (Ed.), *Scenario-based design: Envisioning work and technology in system development* (pp. 59–83). New York: John Wiley & Sons.
- Rogers, E. M. (1995). *Diffusion of innovations* (4th ed.). New York: Free Press.
- Villegas, H., & Williams, M. G. (1997). Benchmarks for workspace awareness in collaborative environments. *International Institute of Informatics and Systemics: Proceedings of the 1997 World Multiconference on Systemics, Cybernetics and Informatics* (pp. 480–486), Caracas, Venezuela.

Endnotes

¹ The identification of any commercial product or trade name does not imply endorsement or recommendation. Groove is a trademark of Groove Networks.

Section VII: Research Method and Empirical Study