UNIVERSITY OF CALGARY

Heuristic Evaluation of Shared Workspace Groupware based on the Mechanics of Collaboration

by

Kevin F. Baker

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

CALGARY, ALBERTA

MAY, 2002

Cite as:
Baker, K. (2002) Heuristic Evaluation of Shared Workspace Groupware based on the Mechanics of Collaboration. MSc Thesis, Department of Computer Science, University of Calgary, Calgary, Alberta, Canada. Available at http://www.cpsc.ucalgary.ca/grouplab/papers/2002/

UNIVERSITY OF CALGARY

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled "Heuristic Evaluation of Shared Workspace Groupware based on the Mechanics of Collaboration" submitted by Kevin F. Baker in partial fulfillment of the requirements for the degree of Master of Science.

_____

Supervisor, Saul Greenberg
Department of Computer Science

_____

Sheelagh Carpendale
Department of Computer Science

_____

Jeff Caird
Department of Psychology

_____

Date

Abstract

Despite the increasing availability of groupware, most systems are not widely used. One main reason is that groupware is difficult to evaluate. In particular, there are no discount usability evaluation methodologies that can discover problems specific to teamwork. In this research study, I adapt Nielsen's heuristic evaluation methodology, designed originally for single user applications, to help inspectors rapidly, cheaply, and effectively identify usability problems within groupware systems. Specifically, I take the Gutwin and Greenberg's (2000) *mechanics of collaboration* and restate them as heuristics for the purposes of discovering problems in shared visual work surfaces for distance-separated groups. As a secondary objective, I revise existing *Locales Framework* heuristics and assess their compatibility with the mechanics. I evaluate the practicality of both sets of heuristics by having individuals with varying degrees of HCI and CSCW expertise use them to uncover usability problems in two groupware systems. The results imply that practitioners can effectively inspect and evaluate groupware with the mechanics of collaboration heuristics where they can identify obstacles to real-time interactions over shared workspaces. The Locales Framework heuristics are not as promising: while inspectors do identify problems inhibiting groupware acceptance, their practicality is limited and they require further improvements.

Publications from this Research

An earlier version of the mechanics of collaboration heuristics (similar to Chapter 3) has appeared in the following peer-reviewed publication:

Baker, K., Greenberg, S. and Gutwin, C. (2001) Heuristic Evaluation of Groupware Based on the Mechanics of Collaboration. In M. Little and L. Nigay (Eds) *Engineering for Human-Computer Interaction*, LNCS Vol 2254, pp. 123-139.

The methodology, analysis, and results from this research study (Chapters 4 and 5) have been summarized in the report listed below. This report has been peer-reviewed and accepted for the upcoming ACM Computer Supported Cooperative Work conference (CSCW 2002).

Baker, K., Greenberg, S. and Gutwin, C. (2002) Empirical Development of a Heuristic Evaluation Methodology for Shared Workspace Groupware. Report 2002-700-03, Department of Computer Science, University of Calgary, Alberta, Canada.

Acknowledgements

I would like to thank my advisor, Saul Greenberg, for your motivation, insight and assistance in getting this research study off-the-ground and out-the-door. Also, thank you for your patience in light of all my personal and professional changes since we first began this journey together. Finally, it has been great getting to know you and the world of CSCW.

My thanks to Carl Gutwin for providing the opportunity to explore new avenues with the mechanics of collaboration. Thanks for your participation and assistance and thanks for the warm hospitality during my short stay in Saskatoon.

My thanks to everybody that helped out along the way. In particular, thanks to all of my heuristic evaluation inspectors, which included the members of GroupLab and the U of S HCI Lab as well as the students from CPSC 591. Also, thanks to my former Nortel colleagues for your valuable feedback with early versions of the heuristics.

Mom and Dad, thanks for everything.

Kaitlyn, thanks for all of the valuable distractions; they kept me relaxed and sane.

Most of all, I would like to thank my wife, Jill. I could not have done it without your support, patience, and understanding. This is for you.

To Jill, with all my love.

Table of Contents

Approval Page ii
Abstract iii
Publications from this Research iv
Acknowledgements v
Table of Contents vii
List of Tables xi
List of Figures xii
List of Figures xii
Chapter 1: Introduction 1
1.1 Difficulties evaluating groupware 2
1.1.1 Social factors 2
1.1.2 Organizational factors 3
1.1.3 Technical factors. 3
1.2 A brief survey of single-user evaluation techniques 4
1.2.1 User observations 5
1.2.2 Field studies 6
1.2.3 Inspection methods 8
1.3 Problems applying single-user techniques to groupware evaluation 10
1.3.1 User observations 10
1.3.2 Field studies 11
1.3.3 Inspection methods 12
1.4 Problem statement and research goals 13
1.5 Research direction 14
1.6 Research overview 14
Chapter 2: Heuristic Evaluation 16
2.1 How to conduct a heuristic evaluation 17
2.1.1 Orientation session 17
2.1.2 Evaluation process 17
2.1.3 Debriefing session 18

2.2 Variations with conducting a heuristic evaluation 20
2.2.1 Presence of the heuristics 20
2.2.2 Scenario vs. self-guided exploration 20
2.2.3 Individual vs. teams 21
2.3 The heuristics 22
2.3.1 Nielsen's original 10 heuristics 22
2.3.2 Evolving the heuristics 23
2.4 Validating the heuristics 24
2.4.1 'Optimum' number of evaluators 25
2.4.2 'Preferred' expertise of evaluators 28
2.4.3 Usability problems uncovered by the heuristics 29
2.5 Conclusion 31
Chapter 3: Groupware Heuristics 32
3.1 Mechanics of collaboration 33
3.1.1 Background 33
3.1.2 Heuristics for supporting the mechanics of collaboration 34
3.1.3 Summary 60
3.2 Locales Framework 60
3.2.1 Background 60
3.2.2 Heuristics for supporting the Locales Framework 61
3.3 Conclusion 62
Chapter 4: Evaluation Methodology 64
4.1 Objective 64
4.2 Pilot study 66
4.2.1 Participants 67
4.2.2 Method 67
4.2.3 Results 67
4.2.4 Discussion 68
4.2.5 Sanity check 69
4.2.6 Locales Framework heuristics 70
4.3 Main research study 70

4.3.1 Participants 70
4.3.2 Materials 71
4.3.3 Method 75
4.3.4 Data collection. 78
4.4 Conclusion 79
Chapter 5: Analysis and Results 80
5.1 Results synthesis 80
5.1.1 Method 81
5.1.2 Results 84
5.1.3 Discussion 85
5.2 Mechanics of collaboration – Analysis and interpretation 86
5.2.1 Relative performance of individual inspectors 87
5.2.2 Performance of aggregates of inspectors 99
5.2.3 Characterizing found teamwork usability problems 102
5.2.4 Usability problems across the two systems 107
5.3 Locales Framework – Analysis and interpretation 108
5.3.1 My personal experience 108
5.3.2 Evaluator feedback 109
5.3.3 Usability problems uncovered 111
5.3.4 Implications 112
5.4 Summary interpretation of results 113
5.5 Discussion 114
5.5.1 Nielsen's methodology 115
5.5.2 Critique of my methodology 118
5.6 Conclusion 120
Chapter 6: Summary 121
6.1 Research goals and summary 121
6.2 Main contributions 123
6.3 Further research 124
6.4 Related Work 127
6.5 Conclusion 127

References 129
Appendix A: Heuristic Evaluation Training Materials 138
A.1 Consent form 139
A.2 Background information questionnaire 141
A.3 Heuristic evaluation instructions 143
A.4 Pre-training feedback form 148
A.5 Mechanics of Collaboration heuristics 155
A.6 Locales Framework heuristics 168
A.7 Problem reports 175
Appendix B: Heuristic Evaluation Usability Problems 176
B.1 Mechanics of Collaboration heuristics – GroupDraw 177
B.2 Mechanics of Collaboration heuristics – Groove 194
B.3 Locales Framework heuristics– GroupDraw 211
B.4 Locales Framework heuristics – Groove 213

List of Tables

Table 3.1 Mechanics of Collaboration heuristics 35
Table 3.2 Locales Framework heuristics 62
Table 5.1 Breakdown of original problem reports after result synthesis 85
Table 5.2 Individual differences in evaluator's ability to find usability problems 87
Table 5.3 Proportion of evaluators who found each GroupDraw problem 93
Table 5.4 Proportion of evaluators who found each Groove problem 94
Table 5.5 Average proportions of problem types found by inspectors 103
Table 5.6 Problems classified according to the mechanics of collaboration heuristics .. 105
Table B.1 GroupDraw Problems reported using Mechanics of Collaboration heuristics 177
Table B.2 Groove Problems reported using Mechanics of Collaboration heuristics 194
Table B.3 GroupDraw Problems reported using Locales Framework heuristics 211
Table B.4 Groove Problems reported using Locales Framework heuristics 213

List of Figures

Figure 2.1 Cheap Shop Order Forms 1 & 2 19
Figure 2.2 Proportion of usability problems in an interface found by heuristic evaluation using various numbers of evaluators (Nielsen and Molich 1990) 26
Figure 2.3 Ratio of benefits to costs for various numbers of evaluators (Nielsen 1994a) 27
Figure 3.1 Overloaded telepointers (Greenberg et al. 1996) 39
Figure 3.2 Scene from TheU – Virtual University (www.ccon.org) 40
Figure 3.3 ClearBoard-2 in use (Ishii and Kobayashi 1992) 41
Figure 3.4 Animation to depict the deletion of an object 46
Figure 3.5 Lock to denote that an object is 'in-use' 49
Figure 3.6 Overview example (Gutwin and Greenberg 1998) 52
Figure 3.7 Fisheye representation (Gutwin 1997) 53
Figure 3.8 Notification Collage 58
Figure 3.9 A snapshot of a TeamWave room (taken from www.teamwave.com) 59
Figure 4.1 Open Registration dialog box and GroupDraw shared workspace 73
Figure 4.2 GroupDraw's Notes functionality 74
Figure 4.3 Groove Outliner tool..... 75
Figure 5.1 Sample of a completed problem report 81
Figure 5.2 Distributions of the number of usability problems found by the evaluators ... 88
Figure 5.3 Proportion of evaluators who found each GroupDraw problem 93
Figure 5.4 Proportion of evaluators who found each GroupDraw problem 94
Figure 5.5 Scatterplot of the proportion of usability problems found by the same evaluators in GroupDraw and Groove 96
Figure 5.6 Problems found by each type of evaluator for both systems..... 98
Figure 5.7 Graph of percentage of problems found by each aggregate of inspectors for GroupDraw and Groove..... 100
Figure 5.8 Average proportion of problems found according to the heuristics 105

Chapter 1: Introduction

Commercial real-time distributed groupware is now readily available due to improvements in hardware, the increased connectivity of the Internet, and the demands of increasingly distributed organizations. Yet with the exception of a few systems (e.g., games and instant messaging), groupware is not widely used. Numerous reasons can be linked to this lack of acceptance. One major problem is that we have a poor idea of what people require of their collaborative systems. We only have sketchy knowledge of how people collaborate in general, and translating what we do know into effective designs is difficult. We also have poor knowledge of specific collaborative tasks. Consequently, developers may not be matching the software they design with the needs of the target user groups. Another serious problem is the lack of practical and inexpensive methodologies for evaluating groupware. As Grudin (1988) points out, many groupware systems are complex and introduce almost insurmountable obstacles to meaningful, generalizable, and inexpensive evaluation. In addition, a host of organizational and political factors that are difficult to characterize and control play a prominent role in determining groupware acceptance (Grudin 1988). All factors contribute to our failure to learn as much as we could from our past experiences of groupware design and evaluation. Consequently, even today's collaborative systems contain serious usability problems that make them awkward and frustrating to use.

Even though Grudin made this point over a decade ago, we have yet to develop and validate techniques that make groupware evaluation cost-effective within typical software project constraints. One way to address this dearth is to adapt accepted evaluation techniques developed for single-user software usability. To that end, my goal is to extend Nielsen's popular heuristic evaluation method to the assessment of real-time, shared workspace groupware. As a focal point for this research study, I look to develop and validate groupware heuristics based upon the mechanics of collaboration (Gutwin and Greenberg 2000). As a secondary and supporting objective, I also explore a complementary set of existing heuristics based on the Locales Framework (Fitzpatrick 1998).

To set this scene, I will first introduce in this chapter the difficulties of evaluating groupware (Section 1.1). Next, I will briefly summarize the many evaluation methodologies now available to practitioners who wish to examine single-user systems (Section 1.2), and will then indicate why it is difficult to apply these methodologies unaltered to groupware (Section 1.3). I then articulate my problem statement, research goals, (Section 1.4) and research direction (Section 1.5). Finally, I present an overview of this thesis.

1.1 Difficulties evaluating groupware

The last decade of human computer interaction (HCI) has seen many evaluation techniques move out of the research arena and into accepted practice. In particular, practitioners now have a toolkit of relatively low cost techniques that lend themselves well to discovering usability problems (refer to Section 1.2). Practitioners also have available to them many 'how-to' books, articles, and training materials e.g. Dumas and Redish (1993), Nielsen and Mack (1994).

While computer supported cooperative work (CSCW) is related to the field of HCI, it has not seen equivalent progress with evaluation methodologies. The problem is that groupware evaluation is considered difficult and costly, especially when compared to the relative ease of evaluating single-user systems. This can be attributed to the increased complexity introduced by social, organizational, and technical factors that influence the adoption of groupware systems.

1.1.1 Social factors

In CSCW, the developer must not only understand human-computer interaction, but also human-human communication and computer-computer interactions. In addition, human-human communication is derived from the social sciences, a body of knowledge not typically known well to developers of groupware. The developer must also address the difficult issues surrounding group dynamics. The study of group behaviour is more complex than that of a single person and requires a different approach (i.e., methodologies based on sociology, management information systems, and anthropology).

The composition and behaviour of groups is continually evolving as a result of its environment thereby changing the group personality. For example, social relationships within the group can alter dramatically within the lifetime of a task or even within a single work session. Group membership and structure can change as members leave and new ones join, further affecting the dynamics. There are also fewer established methods for evaluating group work. Consequently, the task of understanding group interactions has been classified as a "wicked problem"; a problem with "no definitive formulation" and which is "not possible to exhaustively enumerate a set of possible solutions" (Fitzpatrick 1999).

1.1.2 Organizational factors

Grudin (1988) emphasizes context as a complicating factor of cooperative work. The idiosyncrasies of social and political organization in the workplace can lead to the rejection of a good piece of software or the acceptance of a bad one. For example, those who benefit from the introduction of a new system may not be the ones that have to do the work in order to keep the system "running". Grudin (1994) illustrates this problem with a distributed project management application that handles all project-related issues such as schedules, resources, and responsibilities. The project manager sees a direct benefit from this product; however, other group members have the added burden of recording information not typically kept on-line. Likewise, Orlikowski (1994) studied groupware implementation and found that the organizational introduction will interact with cognitive and structural elements in a workplace. These affect the way the software is adopted, understood, and utilized by the users. From an organizational standpoint, groupware systems must also take into account conflict and power relationships. Finally, it is difficult to measure the benefits of groupware to an organization. Thus, many researchers advocate that the group of end users in the context of their workplace must be considered when evaluating groupware.

1.1.3 Technical factors.

Group interfaces are more complex as compared to single-user interfaces since they depict group activity and are controlled by multiple users rather than a single user. As a

result, these interfaces introduce design problems not presented by single-user interfaces. For instance, a basic problem with designing groupware interfaces is how to manage complexity. Multiple users can produce a higher level of activity and a greater degree of concurrency than single users, and the interface must support this complex behaviour (Ellis et al. 1991). This provides challenges for implementing concurrency control measures and architecting the system (e.g., centralized vs. replicated) (Greenberg and Marwood 1994). In terms of evaluation this means that the decisions made on technical grounds can have unintended and perhaps unpredictable effects on the interface and how it is used.

All these issues make the evaluation of collaborative applications a complicated and time-consuming task, typically requiring lengthy field studies and/or elaborate controlled experiments (Pinelle 2000). Since there is no set of low cost evaluation methodologies readily available to groupware researchers and practitioners, an effort should be made to identify new evaluative techniques in order to make evaluation more practical in terms of time and cost. One possible avenue for tackling this problem is the adaptation of single-user evaluation techniques to groupware applications.

1.2 A brief survey of single-user evaluation techniques

Research in HCI has developed a multitude of evaluation techniques for analyzing and then improving the usability of conventional single user interfaces. Each methodology highlights different usability issues and identifies different types of problems; therefore, evaluators can choose and mix an appropriate technique to fit the needs and nuances of their situation (McGrath 1996). There are three primary categories that have been used to distinguish these different types of methods: *user observations*, *field studies,* and *interface inspections*. This section describes some techniques that fall under each category. It is not intended to be a detailed and complete discussion of all evaluation methodologies, but rather to provide an introductory level overview.

1.2.1 User observations

Techniques in this category are conducted in a lab, ideally using a representative sample of the eventual users performing tasks that depict how the product will be used in the "real" world. Evaluators uncover problems, called 'usability bugs', by observing the participants completing the tasks with the interface under evaluation. User observation methodologies include *controlled experiments* and *usability testing*.

Controlled experiments. Controlled experiments are used to establish a cause-and-effect relationship; it must be shown with certainty that the variation of experimental factors (i.e. the independent variables), and only those factors, could have caused the effect observed in the data (i.e., the dependent variable). Rigorous control is used in these experiments to ensure that all other uncontrolled variables (i.e., confounding variables) do not affect the results and their interpretation. To accomplish this goal, participants are divided into comparable groups, usually through random assignment, and the independent variable is applied to one or more of the groups. Each experimental design is then coupled with the appropriate statistical test in order to separate the variability of the subjects from the variability due to the treatment of interest. The statistical analysis allows you to make inferences concerning causal relationships between the manipulation of the independent variables and measured changes in the dependent variables. The statistical analyses are interpreted with regard to the hypotheses developed during the problem definition stage.

Usability testing. As opposed to entailing rigorous controls in order to establish a cause-and-effect relationship, the goal of usability testing is to identify and rectify usability deficiencies. This is in conjunction with the intent to create products that are easy to learn, satisfying to use, and that provide high utility and functionality for the users (Rubin 1994). There is also more flexibility in what portion of the interface can be tested. Designers can manipulate the design of the product to allow them to see how particular features encourage or discourage usability. Participants can do several perhaps unrelated tasks that allow an evaluator to see how the human computer system performs over a broad set of expected uses. The product itself can be changed as the test progresses. For

example, if pilot testing reveals certain problems, then the product can be modified midway to correct them.

When the product is tested with one individual, the participant is encouraged to think aloud during the test. This involves users talking out loud while they are performing a particular task in order to reflect cognitive processes. An evaluator observes the participant performing the task in question by focusing on occurrences such as errors made and difficulties experienced. The information collected can then be applied to remedy the observed usability problems by going through another design iteration of the product, eventually leading to another usability test.

1.2.2 Field studies

A significant problem with performing an evaluation within the laboratory is the failure to account for conditions, context, and tasks that are central to the system's real world use. Part of this failure stems from the fact that many developers of systems often have only partial or naïve knowledge of the "real world" setting where the end system will be used. Field studies allow us to study systems in use on real tasks in real work settings, and to observe or discover important factors that are not easily found in a laboratory setting. Two field study techniques are *ethnography* and *contextual inquiry*.

Ethnography. Ethnography is a naturalistic methodology grounded in sociology and anthropology (Bentley et al. 1992, Hughes et al. 1994, Randall 1996). Its premise is that human activities are socially organized; therefore, it looks into patterns of collaboration and interaction. Randall (1996) stresses four features of ethnography that make it distinct as a method:

1. Naturalistic: involves studying real people and their activities within their natural environment. Only by studying work under these circumstances can one rightfully inform the system's design.
2. Prolonged: it takes time to form a coherent view of what is going on especially for a complex domain.

3. Seeks to elicit the social world from the point of view of those who inhabit it: the appropriate level of analysis is the significance of the behaviour and not the behaviour itself.
4. Data resists formalization: the methodology stresses the importance of context; therefore, there is no 'right' data to be collected.

Data is gathered by the ethnographer observing and recording participants in their environment as they go about their work activities using the technology and tools available to them. This includes focusing on social relationships and how they affect the nature of work. To understand what the culture is doing, the ethnographer must immerse oneself within the cultural framework. The goal of an ethnographic study for system design is to identify routine practices, problems, and possibilities for development within a given activity or setting. The data gathered usually takes the form of field notes but can be supplemented by audio and video data.

Contextual inquiry. To facilitate designing products, contextual inquiry employs an interview methodology to gain knowledge of what people do within their real world context (Holzblatt and Beyer 1996, 1999). Specifically, this is accomplished by first conducting interviews through observations and discussions with users as they work. Target users are representatives of those for whom the system is being developed. Additionally, they should cover a breadth and depth of variation. During the interview, contextual interviewers record what users do and say, their breakdowns and any workarounds, and the artifacts used and how they help get the job done. The objective is to gain an understanding of the users' motivations and strategies. After each interview, team interpretation sessions are conducted whereby each individual interviewer revisits the interview with the rest of the cross-functional team in order to interpret what is going on. From this analysis, representations of the work are created as well as a shared view of the customer. Once all of the interviews are complete and have been individually analyzed, the summary analysis is performed. All the observations from all the interviews are structured using an Affinity diagram in order to produce consolidated versions of the work models. The Affinity diagramming process involves clustering related items into groupings for the purposes of aiding the design process.

1.2.3 Inspection methods

Inspection methods have evaluators 'inspect' an interface for usability bugs according to a set of criteria, usually related to how individuals see and perform a task. These methods use judgement as a source of feedback when evaluating specific elements of a user interface (Mack and Nielsen 1994). Inspection techniques include *heuristic evaluations, task-centered walkthroughs, pluralistic walkthroughs,* and *cognitive walkthroughs*.

Heuristic evaluation. Heuristic evaluation is a widely accepted discount evaluation method for diagnosing potential usability problems in user interfaces (Mack and Nielsen 1994, Nielsen 1992,1993, 1994a,b). With this methodology, a small number of usability experts visually inspect an interface and judge its compliance with recognized usability principles (the "heuristics") (Nielsen 1992, 1993, 1994a). Heuristics are general rules used to describe common properties of usable interfaces (Nielsen 1994a). During a heuristic evaluation, heuristics help evaluators focus their attention on aspects of an interface that are often trouble spots, making detection of usability problems easier. Non-compliant aspects of the interface are captured as interface bug reports, where evaluators describe the problem, its severity, and perhaps even suggestions of how to fix it. Through a process called results synthesis, these raw usability problem reports are then transformed into a cohesive set of design recommendations that are passed on to developers (Cox 1998).

Cognitive walkthroughs. Cognitive walkthroughs (Wharton 1994) are intended to evaluate the design of an interface for ease of learning, particularly by exploration. This is an extension of a model of learning by exploration proposed by Polson and Lewis (1990). The model is related to Norman's theory of action that forms the theoretical foundation for his work on cognitive engineering (Norman 1988). Cognitive walkthroughs also incorporate the construction-integration model developed by Kintsch (1988). These ideas help the evaluators examine how the interface guides the user to generate the correct goals and sub goals to perform the required task, and to select the necessary actions to fulfill each goal.

The designer or an expert in cognitive psychology performs the walkthrough. This expert works through the design for a particular task, step by step, identifying potential problems against psychological criteria. For each task, the expert considers the following: what impact will the interaction have upon the user? what cognitive processes are required? and what learning problems may occur? For each action, information related to the users' goals, tasks and subtasks, knowledge, the visible state of the interface, and the relations and changes among these are recorded in forms. The goal is to identify problematic tasks and task sequences that may break the goal-action stream and cause difficulties in learning the system.

Task-centered walkthroughs. The task-centered walkthrough is a discount usability variation of cognitive walkthroughs. It was developed as one step in the task-centered design process (Lewis and Rieman 1994). This process looks to involve end users in the design process and provide context to the evaluation of the interface in question. First, designers determine who is going to use the system and discuss with them what they do. Next, concrete, detailed examples of tasks are articulated. These tasks will describe what the user wants to do but does not say how the end user would do it. To put these tasks in context, design-specific scenarios for each sample task are constructed. The scenario describes what a user would have to do on a given system interface and what they would see step-by-step in the system as they perform each task. End users validate both the tasks and scenarios for their accuracy. Finally, evaluators use these tasks and the descriptions of the users to step through the scenarios, where at each step they question whether the user has the knowledge to do that step and whether it is believable that the user would actually do it. If the answer to either question is no, then a usability bug has been found. The evaluator marks it, assumes it is solved, and goes on to the next step.

Pluralistic walkthroughs. Pluralistic walkthroughs (Bias 1994) are meetings where users, developers, and human factors people step through an interface for the purposes of identifying usability problems. A pre-defined scenario dictates the participants' interaction with the interface. The scenario ensures that the participants confront the screens just as they would during the successful conduct of the specified task online. The

walkthrough begins when the participants are presented a hardcopy snapshot of the first screen they would encounter in the scenario. Participants are asked to write on the hardcopy of the first panel the actions they would perform while attempting the specified task. After all participants have written their independent responses, the walkthrough administrator announces the "right" answer. The participants verbalize their responses and discuss potential usability problems due to "incorrect" answers. This process continues until the scenario is completed.

1.3 Problems applying single-user techniques to groupware evaluation

Even though CSCW is related to the field of HCI, the standard HCI methodologies have problems when we try to apply them verbatim for the purposes of evaluating groupware. Earlier, I touched upon the difficulties with evaluating groupware in general. Now I will explain how some of these factors have a negative impact on using HCI techniques to assess groupware usability.

1.3.1 User observations

Applying user observation techniques to evaluate groupware is problematic because the experimenter now observes how groups use the system. In contrast to single-user studies, observing groups is difficult. It requires many subjects, which adds the logistical burden of finding and scheduling them. Groups also vary greatly in composition and in how its members interact, which makes observations difficult to analyze and generalize when compared to single-user observations. In addition, experiments involving group work tend to be longer then their single-user equivalents. First, group work requires participants to 'settle down' and develop a rapport. Second, a group's interaction style that comes about with the introduction of groupware unfolds over days and sometimes even weeks (Grudin 1988). While many reasonable single-user studies base their analysis on the first 15 minutes of use, this is not as useful for groupware. Longitudinal studies of system adoption can overcome this problem; however, this increases the cost and time of performing the evaluation, to the point where it may not be done. Introducing time restrictions, as a means to curb time and money expenses, will reduce the accurate

identification of dynamic issues such as embedding into existing environment, learning curves, and acculturation.

Another problem is that user observations in a controlled environment can be too sterile for testing groupware, as it does not take into account the social, organizational, political, and motivational factors that influence how the group accepts and uses the system. Evaluating the application independent from its context of use may provide results that do not generalize well to the real-world situation (i.e., poor ecological validity). Plus, groups are resilient in adapting their behaviours to different controlled situations, which introduces a variable hard to control and measure. Finally, choosing a suitable task in a controlled experiment is difficult since it must encourage both individual and group work.

1.3.2 Field studies

Field studies help solve the contextual problem since the evaluators study people interacting within their workaday world. However, evaluation in the field is remarkably complex and expensive in terms of time, logistics, and analysis (although, Hughes et al. (1994) argue the merits of a quick and dirty approach to ethnography in response to the time and budget pressures). Yet even so, evaluators require experience and a considerable amount of time if their evaluation is to be effective. The number of people that must be observed at each site is high, which can make this task overwhelming. If a new system is being evaluated, it must be robust if it is to be used realistically. Finally, these methodologies have a limited scope of application since they work best at the beginning of design (to uncover and articulate existing work practices) and at the end (to evaluate how systems already deployed in the work setting are used). They are rarely appropriate for iterative design since they are not suited to rapid prototype evaluation. They are also difficult to apply to the design of 'innovative' systems that introduce new or altered work practices: ethnography can only evaluate what is, not what could be.

To overcome some (but not all) of these issues, a 'discount' version of ethnography, design ethnography, has been developed (Salvador and Mateas 1997). Observations and analyses stemming from employing this methodology are used to drive the system design rather than to understand a culture. In addition, this version can be

conducted within a shorter span as compared to traditional ethnographic techniques. However, this method is not yet widely disseminated, accepted, or validated.

1.3.3 Inspection methods

As in single-user applications, groupware must effectively support task work. However, groupware must also support *teamwork*, the 'work of working together'. Inspection methods are thus limited when we use them 'as-is', for they do not address the teamwork components necessary for effective collaboration with groupware. For example, Nielsen lists many heuristics to guide inspectors, yet none address 'bugs' particular to groupware usability. Similarly, a cognitive walkthrough used to evaluate groupware gave mixed and somewhat inconclusive results (Erback and Hook 1994). Other researchers are providing a framework for typical groupware scenarios that can form a stronger basis for walkthroughs (Cugini et al. 1997).

I speculate in this research study that some of these inspection techniques can be altered to evaluate groupware. Specifically, I chose to adapt Nielsen's heuristic evaluation methodology since it is popular with both researchers and industry for several important reasons. It is low cost in terms of time since it can be completed in a relatively short amount of time (i.e., a few hours). End-users are also not required; therefore, resources are inexpensive. Because the heuristics are well documented and worked examples have been made available (e.g., Nielsen 1994a, b), they can be easy to learn and apply. Also, heuristic evaluation is becoming part of the standard HCI curriculum (e.g., Greenberg 1996) and thus known to many HCI practitioners. Non-usability experts can also use this technique fairly successfully (Nielsen 1994a). As well, it is cost-effective: an aggregate of 3-5 usability specialists will typically identify ~75% of all known usability problems for a given interface (Nielsen 1994b). All these factors contribute to the significant uptake of heuristic evaluation in today's industry since this technique can be easily and cost-effectively integrated into existing development processes while producing instant results. In expanding heuristic evaluation for the purposes of evaluating groupware, I look to capitalize on all these factors that make this methodology a success.

1.4 Problem statement and research goals

The motivation behind this research is that current real-time distributed groupware systems are awkward and cumbersome to use, a situation partly caused by the lack of practical groupware evaluation methodologies. My general research goal is to develop and validate a groupware evaluation methodology that is practical in terms of time, cost, logistics, and evaluator experience, while still identifying significant problems in a groupware system. To narrow the scope, I adopt an existing discount usability technique to *real-time, distributed groupware* supporting *shared workspaces*. Real-time distributed groupware encompasses collaborative systems that enable multiple people to work together at the same time but from different locations. A shared workspace is "a bounded space where people can see and manipulate artifacts related to their activities." (Gutwin 1997). This application genre is very common (e.g., real-time systems for sharing views of conventional applications). Specifically, I focused on heuristic evaluation as this methodology in its current state satisfies the practicality criteria of time, cost, logistics, evaluator experience while still identifying significant problems in a single user systems. I believe that this technique and its strengths can be extended to assessing collaborative systems.

From this general research goal, my specific research sub-goals follow.

1. I will propose a new set of heuristics that can be used within the heuristic evaluation methodology to detect usability problems in real-time, distributed groupware with a shared workspace.
2. I will demonstrate that the adapted heuristic evaluation for groupware remains a 'discount' usability technique by analyzing the ability of inspectors to identify problems in collaborative applications.

Developing the heuristics will draw upon past research that has identified criteria necessary for the development of 'good' real-time, distributed groupware supporting shared workspaces. Demonstrating the methodology as a 'discount' usability technique enforces the notion that the heuristics are easy to learn and can subsequently be used by individuals that are not necessarily experts in groupware. In validating the groupware

heuristics, I am demonstrating that these heuristics do in fact identify a category of usability problems inherent to groupware acceptance.

1.5 Research direction

At this point, I need to elaborate on the circumstance and my resulting decisions that led to the main thrust of my research study: to derive and validate groupware heuristics based on the mechanics of collaboration. The purpose is to provide insight into my disproportionate focus with the Locales Framework heuristics.

My original objective was to build upon Greenberg et al.'s (1999) preliminary work on the Locales Framework heuristics. While conventional heuristics are easy to learn and apply, an outstanding concern from the original study was that heuristics based on the Locales Framework are complex, which in turn might require a greater level of evaluator training and experience. To that extent, I set out to assess these heuristics by studying how well inspectors unfamiliar with the Locales Framework were able to apply these heuristics to identify usability problems in groupware systems. Shortly afterwards Gutwin and Greenberg (2000) introduced the mechanics of collaboration framework. This framework was created with low-cost evaluation methods for groupware in mind; therefore, I decided to refocus my research in this direction. Subsequently, this research study concentrates on creating and validating the mechanics of collaboration heuristics. While I still explore the locales framework heuristics, they are not my primary area of interest and hence I have devoted less time and effort in this study to their validation.

1.6 Research overview

Chapter 2 chronicles Jakob Nielsen's design, validation, and subsequent evolution of his original 10 heuristics for the purposes of evaluating single-user interfaces. Since I look to piggyback on his methodology and loosely replicate how he validated it, I believe it is necessary to provide a brief history on how the existing heuristic methodology was developed, validated, and updated.

Chapter 3 describes in detail the eight heuristics derived from Gutwin and Greenberg's (2000) *mechanics of collaboration* framework. These heuristics form the basis for the rest of the research. In addition, five complementary heuristics evolving from the *Locales Framework* (Fitzpatrick 1998) are also briefly introduced.

Chapter 4 details the two-step methodology I employed to validate the groupware heuristics as a discount usability method for groupware. First, a pilot study was conducted to review and subsequently improve the heuristics. Next, two categories of inspectors with varying levels of expertise in HCI and CSCW used the revised heuristics to evaluate two groupware systems. The resulting problem reports form the raw data for the forthcoming analysis.

Chapter 5 describes the results synthesis process employed to transform the inspectors' raw problem reports into a consolidated list of usability problems for each groupware system. Next, I systematically analyze these lists to derive conclusions regarding the practicality of both sets of groupware heuristics. Finally, I discuss some of the factors affecting my results and how I interpret these results

Chapter 6 summarizes how the goals of my research have been satisfied and the contributions made. In addition, I look to the future and discuss what still needs to be done to help evolve the heuristics for the purposes of developing a robust and effective low-cost technique for evaluating groupware.

Chapter 2: Heuristic Evaluation

Many obstacles to getting usability results during software development can be attributed to the lack of 'usable' usability methods (Nielsen 1995). Consequently, many developers abstain from practicing usability engineering since they view these methodologies as time consuming, expensive, and complex. To address this issue, the "discount usability engineering" approach was developed. Its goal is to help evaluators find major usability problems in a user interface design without requiring a large set of resources – time, budget, staff, and expertise.

Nielsen's heuristic evaluation is a well-known and well-established technique within discount usability engineering. It uses simple and approximate methods, instead of the more formal and exact methods of other methodologies (see previous chapter for details on these other methodologies). Nielsen acknowledges that heuristic evaluation may produce discount results that are not as precise or reliable as other methods. However, the more careful methods have the drawback of being more expensive in terms of resources. Therefore, the 'simpler' heuristic evaluation stands a much better chance of actually being used in practical design situations. Performing some usability engineering work, even though the method employed is not 'the best' and the results are not 'perfect' is always better than doing none at all.

What makes traditional heuristic evaluation an effective methodology for evaluating single-user interfaces are the same reasons why I chose to expand this technique to groupware evaluation. My goal is to develop and validate a set of groupware heuristics as a tool for discount usability engineering. Consequently, I replicate and adapt to a groupware setting many of Nielsen's processes that he used to devise, validate, and revise his heuristics. To set the scene, this chapter provides a summary of Nielsen's processes. I begin by describing the typical process used by inspectors to conduct heuristic evaluations (Section 2.1) as well as variations to this process (Section 2.2). Next, I discuss how Nielsen developed the original set of heuristics (Section 2.3) and how he subsequently revised his heuristics into a better set (Section 2.3). Finally, I detail his

processes for validating this method as a discount usability engineering technique (Section 2.4).

2.1 How to conduct a heuristic evaluation

In Section 1.2.3 of the previous chapter, I outlined the basics of heuristic evaluation. Here, I provide an overview of the heuristic evaluation process, which typically has three stages: *orientation*, *evaluation process*, and *debriefing session*.

2.1.1 Orientation session

As its name suggests, the purpose of the orientation session is to provide the interface inspectors information on the major components of a heuristic evaluation.

1. The method: Evaluators who are not UI specialists require an introduction to the heuristics and the methodology.
2. The domain: To increase the evaluators' domain knowledge, they are given a short lecture of the system under test.
3. The scenario (optional): If the evaluators are not domain experts, they can be presented with a scenario (e.g., sample tasks and how the interface would be used to perform them) to help explore and evaluate the system.

2.1.2 Evaluation process

In principle, the evaluators will decide how they want to proceed with evaluating the user interface. Nielsen (1994a) recommends they go through the interface at least twice. The first pass helps the evaluator get a feel for the "flow" of the interaction as well as the general scope of the system. This provides a further understanding of the domain, especially if they are not experts in it. The second pass allows the evaluator to focus on specific interface elements while knowing how they fit into the bigger picture.

As the evaluators independently proceed through the interface, they inspect the various dialogue elements and compare them with the list of heuristics. Heuristics are general rules used to describe common properties of usable interfaces, e.g., such as 'Provide Feedback', and will be described more fully in Section 2.3 (Nielsen 1994a). The

heuristics help focus the evaluator's attention on aspects of an interface that are typically sources of trouble to assist with the detection of usability problems. When an evaluator uncovers within the interface, in their opinion, a non-compliance with a heuristic, both the usability problem and the heuristic violated are recorded. Results are recorded either as written problem reports from each evaluator or by having the evaluators verbalize their comments to an observer as they go through the interface (Nielsen 1994a). The end result is a list of usability problems in the interface with references to the violated heuristics.

Understanding heuristic evaluation is often easier by example rather than by explanation. To assist the reader, I include on the next page a sample interface and sample problems detected through a heuristic evaluation with Nielsen's (1993) original heuristics (with the exception of 'Help and Documentation').

2.1.3 Debriefing session

Finding usability problems and analyzing them are two different processes, which should not necessarily be combined into a single evaluation session. Nielsen (1994a) found that evaluators were not generally adept at stating which heuristic was violated by each usability problem or classifying the severity of the problem during the evaluation session. Instead, they were more focused on inspecting the interface and finding new usability problems. Consequently, Nielsen (1994a) recommends a debriefing session after all evaluations are complete. Participants should include the evaluators, any observers used during the evaluation sessions, and representatives of the design team. The debriefing session is conducted primarily in a brainstorming mode. During this session, the participants achieve a consensus on the best way to group together all duplicate problem reports. The next step is to discuss severity ratings and eventually evolve the problem report groupings into a list of solutions or recommendations to improve the interface under evaluation. Typically, generating fixes to the observed usability problems is straightforward since each problem is referenced to the violated heuristic, which is based on established usability principles (Nielsen 1994a). Cox (1998) has labeled this portion of the heuristic evaluation methodology, results synthesis, and has provided a more detailed methodology of how this step can be done effectively.

Example: Heuristic Evaluation of the Cheap Shop Interface

The Cheap Shop Department Store is a catalog-based store. Shoppers in the store browse a paper catalog for items and then purchase items by filling out the screens below.

- To create an order, shoppers enter their personal information and their first order on form 1.
- To create more orders, shoppers click the 'Next Catalog Item' button to go to form 2.
- To complete the order(s), shoppers click the 'Trigger Invoice' button. The system automatically tells shipping and billing about the order and returns to a blank form 1.
- To cancel an order, shoppers do not enter any values for 30s (as if they walk away). The system will then clear all screens and return to the main screen.
- Input fields are checked by the system when a button is pressed. Erroneous fields will blink for 3s and then be cleared by the system. The shopper can then re-enter the correct values in those fields.

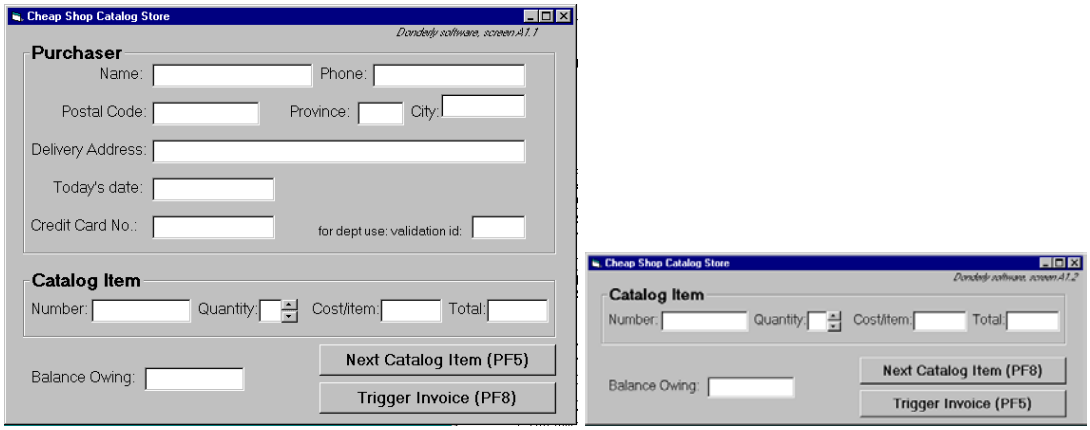

Figure 2.1 Cheap Shop Order Forms 1 & 2

Usability Problems

Simple and Natural Dialogue – The fields for entering the shopper's personal information should be in a natural order. Typically, an address is entered by delivery address, city, province, and then postal code.

Speak the User's Language – The button labeled 'Trigger Invoice' does not use common terminology for a shopper who wants to complete and submit the order.

Minimize User Memory Load – The text field for entering the telephone number should make it obvious to the user what the accepted format for entering this information into the system.

Consistency – The hotkeys for the identical buttons on each form are different.

Feedback – The shopper does not receive feedback that an order has been successfully completed and sent to shipping and billing.

Clearly Marked Exits – There is no clearly marked avenue for canceling an order that has been started. It is not obvious to the shopper that (s)he must not enter any values for 30 sec. to cancel the order

Shortcuts – The system should automatically fill-in today's date as opposed to forcing the shopper to perform this action.

Good Error Messages – Blinking erroneous fields for 3 sec. and then clearing them does not provide the shopper with any information regarding the problem and how to correct it.

Prevent Errors – Instead of error messages for incorrect telephone numbers, the system could accept common ways of entering a number (e.g. placing parentheses around the area code).

2.2 Variations with conducting a heuristic evaluation

There are a series of variations in the way a heuristic evaluation can be conducted:

- presence of the heuristics
- scenarios vs. self-guided exploration
- individuals vs. teams

2.2.1 Presence of the heuristics

The presence of the heuristics during the evaluation will depend upon the inspector's preference. Karat et al. (1992) found that experts did not need the heuristics in front of them during the evaluation. They were already familiar with the concepts due to their experience with graphical user interfaces and other systems. However, Karat et al. thought less experienced users would find it useful having them present. As well, evaluators should not limit themselves to reporting only those problems that match a particular heuristic. Any additional usability problems noticed but not explicitly addressed by a heuristic should also be recorded.

2.2.2 Scenario vs. self-guided exploration

When evaluators perform a heuristic evaluation, they can either step through the interface using representative end user tasks (i.e., scenarios) or self-guided exploration. Each approach has its advantages and disadvantages. As in walkthroughs, scenarios are intended to represent how the system will be used by the end-users. They list the various steps a user would take to perform a set of realistic tasks. Scenarios are constructed on the basis of a task analysis, involving consultations with end users and observation of these users in their work setting. The main advantage of scenarios is to help supply evaluators, with little domain expertise, the knowledge needed to operate the system (Karat 1994, Nielsen 1993). Plus, the use of well-established scenarios can provide some assurance that real world problems will be identified (Nielsen 1992). Scenarios can also be constructed to ensure that specific features of interest to the usability specialists are evaluated. This is useful if certain areas of a system require more attention than others. However, limiting the scope of the evaluation to specific features in an interface reduces

the evaluators' range of exploration. As a result, potential usability problems may be missed. Finally, within a real situation users will not necessarily be given explicit task scenarios.

Without scenarios to aid in the evaluation of a user interface, evaluators will use self-exploration. Typically, this involves generating their own goals with meaningful tasks to achieve them (Karat 1994). This depends on how much the inspectors are encouraged to explore and how task-oriented they are (Karat 1994). This freedom allows the evaluators more opportunities to explore more diverse aspects of the interface since they are not limited to the interface features that are focused on by the scenarios.

Research suggests that walkthrough groups favoured the use of scenarios in finding usability problems (Karat et al. 1992).

2.2.3 Individual vs. teams

Are heuristic evaluations more effective when conducted individually or in teams? Nielsen (1993) recommends performing heuristic evaluation with each evaluator inspecting the interface independently. In turn, the individual evaluator is presented the opportunity to form his/her own evaluation about an interface without being influenced by others. This ensures results from each evaluator are independent and unbiased with greater variability in the kinds of errors found. Plus, with single evaluators there is no overhead required to organize group meetings. Only after all evaluations have been completed are the evaluators allowed to communicate and have their findings aggregated.

However, Sawyer et al. (1996) suggest multiple evaluators looking at the interface together because they will find more problems and suggest better solutions than evaluators working alone. People can reinforce each other and help each other to notice problems that may go unnoticed individually. Also, multiple evaluators help to filter out problem predictions that are not plausible end-user problems (Mack and Nielsen 1994). Sawyer et al. (1996) recommends a maximum of 2 evaluators during a heuristic walkthrough. They found that more than two causes a bottleneck: while the observer takes notes, the other evaluators risk becoming bored.

A comparison of heuristic evaluations performed individually against those done in pairs suggests that teams achieve better results than individuals in some areas (Karat

et. al. 1992). Specifically, teams identified more usability problems; however, the total number of "significant problem areas" was consistent across the two types.

2.3 The heuristics

2.3.1 Nielsen's original 10 heuristics

Nielsen and Molich (1990) developed their original set of heuristics from several years of experience teaching and consulting within usability engineering. They chose heuristics from their understanding of typical problem areas of usability, as well as an informal consideration of existing interface guidelines (Nielsen and Molich 1990, Molich and Nielsen 1990). They developed heuristic evaluation with the goal of making its methodology easy to teach (Nielsen 1994b).

As a starting point, Nielsen and Molich (1990) suggest that a good set of heuristics should be small (e.g., around 10). This helps the inspectors to remember and reference them, while still being rich enough to detect a large number of usability problems typically found in user interface designs. In terms of guidelines, these heuristics can be considered a small set of very general design principles. This level of generality means that heuristics are motherhood statements. They facilitate the detection of usability problems by focusing the inspector's attention on portions of an interface that are often sources of trouble. An evaluator must exercise some judgment and interpretation skills to uncover particular problems. That is, they are not a checklist.

Nielsen's original set of 10 heuristics is listed below. Of course, novice evaluators are expected to require more training as to the nuances of these heuristics. For example, Nielsen (1993) gives a detailed explanation of each heuristic in Chapter 5 of his book.

1. Simple and natural dialogue
2. Speak the users' language
3. Minimize user memory load
4. Consistency
5. Feedback
6. Clearly marked exits
7. Shortcuts
8. Good error messages
9. Prevent errors
10. Help and documentation

2.3.2 Evolving the heuristics

Since the introduction of the original set of heuristics in 1990, they have been refined due to contributions from other guidelines and knowledge gained from performing heuristic evaluations. As a means to "synthesize a new set of usability heuristics that is as good as possible", Nielsen (1994b) studied how well 101 usability principles accounted for 249 usability problems. These usability principles were chosen from his original set of heuristics as well as from six other collections of published principles or guidelines, while the usability problems were collected from 11 projects evaluated by either heuristic evaluation or user testing. A formal factor analysis and an examination of the explanatory coverage of each heuristic were then performed.

The factor analysis involved rating each heuristic on how well it explained each usability problem. From the analysis, seven factors were developed which accounted for the most variance in the usability problems (a factor was a summary of the underlying usability phenomenon covered by a group of related heuristics). Unfortunately, the seven factors only added up to 30% of the total variance. In other words, usability problems are due to a broad variety of underlying phenomena as opposed to a small set of usability factors. Due to the results of the factor analysis, Nielsen was forced to find a set of usability heuristics that accounted reasonably well for the majority of the usability problems. With this goal, the explanatory coverage possible by various combinations of the existing heuristics was determined. From this analysis, Nielsen found that a manageable number of heuristics (10) accounted for the majority of the total number of usability problems. Similarly, 10 heuristics explained most of the usability problems with major severity ratings. Almost all of the seven aforementioned usability factors were represented by these two sets of heuristics. Given this overlap, the seven factors were used as the backbone for the new set of heuristics. To round out the revised set of heuristics, Nielsen included two more heuristics that covered a wide number of the total usability problems. He felt that these last two were important based on his experience. The following are Nielsen's revised set of ten heuristics.

1. Visibility of system status
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose, and recover from errors
10. Help and documentation

In addition, Muller et al. (1996) introduced two extensions beyond Nielsen's original technique. These were included to favour users as direct participants, and to add process-oriented heuristics. Empirical justification for the extensions involved applying 13 new heuristics (Nielsen's 10 revised heuristics and Muller's 3 process-oriented ones) to a prototype. Analyzing the percentage of potential usability problems identified by each heuristic and the percentage of recommendations based on each heuristic revealed problems that would have remained undetected without the additional heuristics. Along with the 3 new heuristics, Muller et al. (1996, 1998) provided theoretical basis for a fourth, thereby increasing the total number of heuristics to 14. The 4 new heuristics are:

1. Respect the user and his/her skills
2. Pleasurable experience with the system
3. Support quality work
4. Protect the user's privacy

2.4 Validating the heuristics

To validate the practicality of heuristic evaluation as a discount usability methodology, Nielsen conducted a series of six case studies with his original 10 heuristics (Molich and Nielsen 1990, Nielsen and Molich 1990, Nielsen 1992, 1994a). Included in the case studies were three paper prototypes and three running systems. Findings were used to draw conclusions regarding the heuristic evaluation methodology in the following areas:

- 'optimum' number of evaluators;
- 'preferred' expertise of evaluators; and
- usability problems uncovered by the heuristics.

2.4.1 'Optimum' number of evaluators

In an attempt to validate the practicality of his heuristics in terms of the human resources required, Nielsen analyzed the 'optimum' number of evaluators that should be employed during a heuristic evaluation. As the reader will see, we cannot expect one evaluator to uncover the majority of usability problems. The real question is "How many inspectors are needed to uncover the majority of the problems?".

Performance of individual evaluators. Single evaluators will typically miss many usability problems during a heuristic evaluation. Based on the aggregate results from the six case studies (Nielsen 1992), single evaluators uncovered approximately:

- 35% of usability problems;
- 42% of major usability problems; and,
- 32% of minor usability problems.

While these results will vary according to the complexity of the interface under test and the expertise of the inspectors, we can expect 'poor' results when relying on a single evaluator to perform a heuristic evaluation. Some may suggest that doing a heuristic evaluation with only a single-evaluator performance is unacceptable for a usability engineering project. This is not necessarily the case, for performing a heuristic evaluation and finding some problems is much better than finding no problems at all! This is especially true with a design schedule that only permits a single inspector to perform a heuristic evaluation. Of course, increasing the number of evaluators and/or supplementing the heuristic evaluation with other usability engineering methods can reveal more problems.

Nielsen and Molich (1990) then compared the individual differences between evaluators performing heuristic evaluations. They found that most evaluators uncover an 'average' amount of usability problems, while a few do very well and a few do rather poorly. This difference in the performance of individual inspectors becomes larger as the interface becomes more difficult to evaluate.

Increasing the number of evaluators. The effectiveness of heuristic evaluations can be significantly improved by involving multiple evaluators, for different people find

different usability problems. That is, there is a substantial amount of non-overlap between the sets of usability problems found by individual inspectors (Nielsen and Molich 1990, Nielsen 1992, 1994a). While some usability problems are found by almost everybody, others are only found by a very few evaluators. Furthermore, one cannot just identify the 'best' evaluator and rely solely on that person's findings. First, the same person will not necessarily be the best evaluator every time. Second, some of the hardest-to-find usability problems are found by evaluators who do not otherwise find many problems. Therefore, Nielsen recommends involving multiple evaluators with any heuristic evaluation.

It must be realized that the 'optimum' number of evaluators is not a reliable magic number and depends on many factors such as budget, schedule, resources, and evaluator expertise. To determine a recommended number of evaluators, Nielsen formed aggregates by choosing the number of people in the aggregate randomly from the total set of evaluators in that experiment (Nielsen and Molich 1990). The proportion of usability problems found given the number of evaluators is shown below in Figure 2.2. The curve represents the average of six case studies of heuristic evaluation (Nielsen 1992).

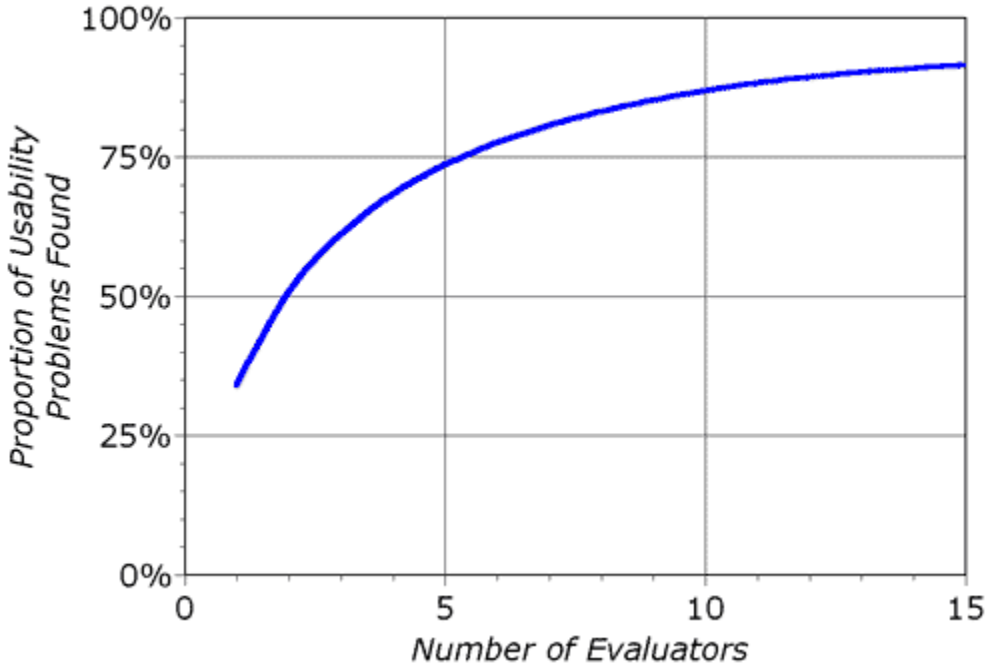

Figure 2.2 Proportion of usability problems in an interface found by heuristic evaluation using various numbers of evaluators (Nielsen and Molich 1990)

The curve rises steeply from one evaluator to about five evaluators, and flattens out somewhat afterwards. Thus increasing the number of inspectors beyond five does not yield an increase in problems detected that is proportional to the added effort. Therefore, Nielsen (1992, 1994a) generally recommends using between 3 and 5 evaluators for best effect, where using five evaluators will find from two-thirds to three-quarters of the usability problems in an interface. This is a considerable improvement when compared to the results from single evaluators.

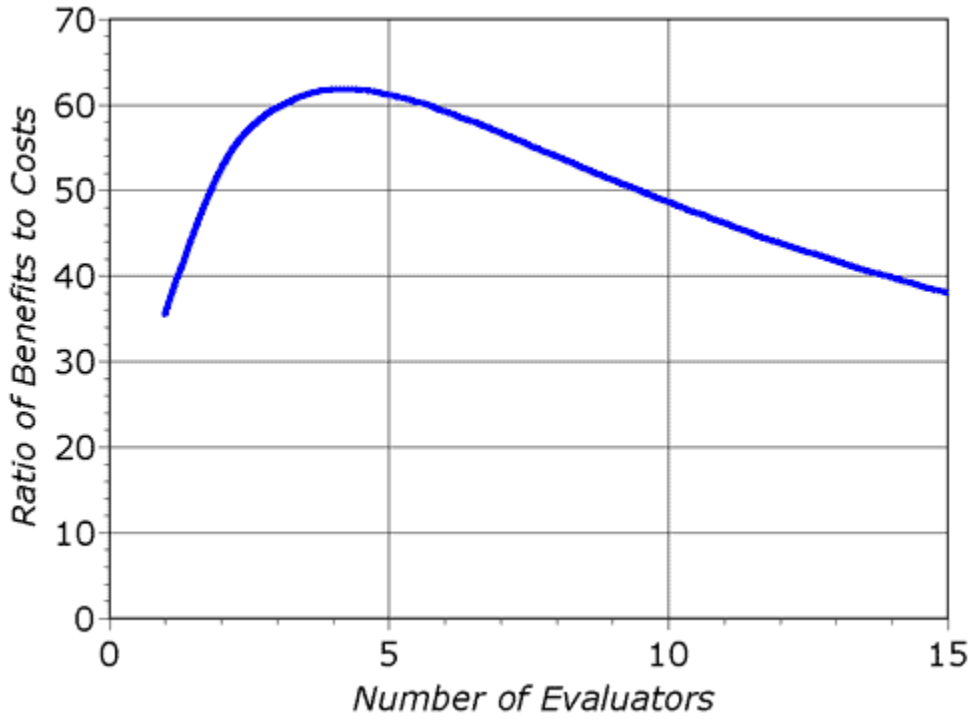

Figure 2.3 Ratio of benefits to costs for various numbers of evaluators (Nielsen 1994a)

In addition to his observations, Nielsen (1994a) formulated a cost/benefit ratio for heuristic evaluations to help determine the optimum number of evaluators. The results of this analysis are illustrated in Figure 2.3. He concluded that the maximum cost/benefit ratio for a medium/large project would be to perform a heuristic evaluation with four evaluators. This supports heuristic evaluation as a discount usability engineering method that can be applied on a modest budget. Ultimately, the optimum number of evaluators to

use will depend upon the compromise made between the increased costs of using each extra evaluator versus the cost of not identifying usability problems.

2.4.2 'Preferred' expertise of evaluators

Heuristic evaluation was originally developed as a usability engineering method for evaluators who had some knowledge of usability principles, but were not necessarily usability experts. However, subsequent research proves that the method is more effective when the evaluators are usability experts. To determine the effects of evaluator expertise on the performance of heuristic evaluations, Nielsen (1992) had three groups of evaluators with varying levels of usability expertise evaluate a paper prototype of a voice response system. These groups included:

1. Novice evaluators were computer science students having completed their first programming course but with no formal knowledge in user interface design. They were considered novices with respect to usability and not computers.
2. Regular specialists were individuals with several years of job experience in user interface design and evaluation and/or graduate degrees in the area. They had no expertise in the particular kind of interface being evaluated (voice response).
3. Double specialists were individuals with expertise with both user interface issues as well as the interface under test.

He found the following individual performances:

- novice evaluators found an average of 22% of the problems;
- regular specialists found an average of 41% of the problems; and
- double specialists found an average of 60% of the problems.

The performance of novice evaluators was fairly poor relative to the other types of inspectors. Regular specialists uncovered more problems due to their expertise with usability issues in general. Double specialists performed even better since they were able to identify those problems specifically related to the domains due to their experience with the usability issues for the kind of user interface under test. Therefore we can conclude that it helps to have usability expertise with respect to the type of user interface being evaluated.

From the previous section we see that Nielsen strongly agrees with the use of aggregates of inspectors during a heuristic evaluation. Yet Nielsen also found that the increased expertise of the regular and double specialists does not offset the need to incorporate multiple inspectors. He formed a large number of random groups and found:

- fourteen novice evaluators is necessary to find more than 75% of the problems;
- three to five regular specialists will find a high proportion of problems; and
- three to four double specialists will find most problems.

One of the main advantages with heuristic evaluation is its low cost. A concern, then, is that evaluators require some experience with the principles to effectively apply them (Nielsen 1992): these could be hard to find and may demand high salary and/or consulting fees. Obviously, it would be preferable to employ usability specialists as evaluators and even better would be to use double specialists. However, as the expertise of the evaluators increases so does the cost of performing a heuristic evaluation. Usability experts may also be hard to come by, especially if they also require expertise in a particular application domain. It is advantageous from a discount usability engineering (i.e. cost) perspective to perform inspections with people having little or no usability expertise. Unfortunately, while it may be easy to find novice evaluators, they will not provide as fruitful results. Given the choice between lower costs or increased productivity, Nielsen recommends using the costlier usability specialists as a minimum since it is believed that heuristic evaluation draws much of its strength from the skilled usability professionals who use them (Jefferies et al. 1991). The importance of these people's knowledge and experience cannot be underestimated. Realistically, trade-offs must be made between the cost of not finding usability problems and the cost of using expert evaluators to perform a heuristic evaluation.

2.4.3 Usability problems uncovered by the heuristics

Nielsen analyzed how evaluators' employed the heuristic evaluation methodology to discover usability problems in an interface. More specifically, he looked at the ability of the heuristics to uncover *major vs. minor usability problems*; *each heuristic's explanatory coverage*; and the *location of problems in the dialogue*.

Major vs. minor problems. Results from the six case studies indicate that heuristic evaluation is a good method for finding both major and minor problems in a user interface. Major usability problems are defined as those that have serious potential for confusing users or causing them to use the system erroneously, whereas minor problems may slow down the interaction or inconvenience users unnecessarily (Nielsen 1992). In terms of absolute numbers, evaluating an interface via heuristic evaluation typically uncovers more minor problems than major ones. On average, inspectors will uncover twice as many minor problems. Despite the fact that inspectors will typically report more minor than major problems, individual inspectors will usually report a higher proportion of the total number of major usability problems in a given interface in comparison to the proportion of minor problems observed. Nielsen concluded that although the evaluators pay more attention to major problems, the minor ones are not neglected.

Explanatory coverage of each heuristic. Heuristic evaluation involves judging interfaces according to established usability principles; therefore, one might expect that it would be easier to find problems violating certain heuristics. For the most part, this is not the case. User interface problems uncovered during a heuristic evaluation have about the same probability of being found by most of the heuristics. However, problems that violate the "prevent errors", "clearly marked exits", and "good error messages" heuristics are seemingly more difficult to identify. If necessary, additional measures can be taken to find problems relating to these heuristics.

Location of problems in dialogue. Nielsen (1992) also looked at whether the circumstances under which the problems could be located had any influence. He stated that usability problems could be located in four different ways:

- in a single dialogue element;
- in two or more locations that have to be compared to find the problem;
- as a problem with the overall structure of the interface; and
- as something that is missing and should be included.

An analysis of the difference between the four categories did not prove to be statistically significant. When evaluating running systems, problems in the 'missing and should be

included' category are slightly easier to find than all other problems. However, this is reversed in paper prototypes where these problems are much harder to find. This suggests that one should look harder for missing dialogue elements when evaluating paper mockups.

2.5 Conclusion

Heuristic evaluation is a widely accepted discount evaluation method for diagnosing potential usability problems in user interfaces (Nielsen and Molich 1990, Nielsen 1992, 1993, 1994a, 1994b). Heuristic evaluation is popular with both researchers and industry. It is low cost in terms of time, resources, and expertise. I look to capitalize on all these factors as I expand this technique to groupware evaluation.

This chapter was intended to provide an overview of the traditional heuristic evaluation, its validation as a discount usability methodology, and how it has been revised and improved. The purpose is to provide insight into the process that I will use to develop and validate the efficiency and practicality of the groupware heuristics. As a first step, two sets of groupware heuristics for evaluating real-time groupware are presented in the next chapter.

Chapter 3: Groupware Heuristics

Nielsen's existing heuristics were derived from how well they explained usability problems resident within single-user systems. However, these heuristics are insufficient for groupware evaluation since they do not cater to groupware usability, which is distinctly different from single-user usability. *Single-user usability* has been defined as the degree to which a system is effective, efficient, and pleasant to use, given a certain set of users and tasks (e.g., Shackel 1990). Within this context, usability emphasizes 'task work': how a person performs the domain tasks and activities that result in the end products like drawings, documents, or models. Groupware must also support task work to proceed effectively, efficiently, and pleasantly; however, these systems must go one step further and support teamwork—the work of working together—in order to be truly usable. Thus we can define *groupware usability* as the degree to which a system supports both single-user usability and teamwork. While this definition is a starting point, we need a better understanding of what we actually mean by 'support for teamwork'. As we will see shortly, this is no easy task: teamwork involves activities ranging from low level mechanical acts necessary for almost any type of real time collaboration, to those that are social and affective in nature.

If we want to apply the heuristic evaluation methodology to groupware, we need new heuristics that identify those interface aspects necessary for effective teamwork. Using these, the inspector can then examine the interface to see if adequate support is provided to a group. The problem is that developing new heuristics to evaluate teamwork is complicated since—unlike the single-user interface literature that helped Nielsen set up his heuristics—there is no broad corpus of design guidelines specific to teamwork issues. As a starting point, I have instead adapted two sources as the basis for two potential sets of groupware heuristics.

1. Mechanics of Collaboration (Gutwin and Greenberg 2000)
2. Locales Framework (Fitzpatrick 1998)

These sources were chosen because they contain some of the few theories or frameworks dealing with teamwork that were specifically created with groupware in mind.

Gutwin and Greenberg's (2000) *mechanics of collaboration* identifies those activities that support the mechanics of how people interact over a shared visual workspace. These activities include group members communicating, providing assistance, coordinating activity, dividing labour, and monitoring each other's work. I believe that heuristics derived from these mechanics can be applied to task-based groupware (e.g., shared whiteboards, brainstorming tools, etc.), which comprise the majority of existing groupware systems today. In contrast, Fitzpatrick's (1998) Locales Framework deals more with the social issues surrounding the use of groupware. Because they define social *versus* mechanical aspects of teamwork, I believe that heuristics derived from the Locales Framework are better suited for evaluating design subtleties of how social interaction is supported in general groupware environments that support a broad variety of tasks (such as Teamwave Workplace) rather than the mechanical aspects of task-specific groupware. While there is some overlap between heuristics suggested by the mechanics of collaboration and by the Locales Framework, they are both inspired by quite different conceptual frameworks. Therefore, I do not view the two sets as competing heuristics, but rather as complementary when uncovering usability problems in groupware.

To explain how we adapted these two sources as heuristics, I divide the rest of this chapter into two main parts. The first part (Section 3.1) provides a brief description of the mechanics of collaboration, and how I adapted them into eight heuristics. The second part (Section 3.2) follows a similar structure; details of the Locales Framework are presented along with a brief description of the five heuristics stemming from this framework.

3.1 Mechanics of collaboration

3.1.1 Background

The mechanics of collaboration frames the low level actions and interactions that small groups of people do if they are to complete a collaborative task effectively. These mechanics are specific to shared workspace groupware. They include communication,

coordination, planning, monitoring, assistance, and protection. Gutwin developed this framework both from his experience building shared workspace systems and how they were used, and from an extensive research of shared workspace usage and theory developed by others (e.g., Bly 1988, Tang and Leifer 1988, Tang 1991, Gutwin 1997, Gutwin and Greenberg 1999).

The underlying theory of the framework is that while some usability problems in groupware systems are strongly tied to social or organizational issues in which the system has been deployed, others are a result of poor support for the basic 'mechanical' activities of collaborative work in shared spaces. It is these basic mechanical activities—not the social or organizational issues—that the framework articulates. Therefore, supporting the mechanics help groups to communicate effectively and efficiently about the task and to smoothly and easily coordinate their actions. Since the mechanics of collaboration are for the most part separate from organizational politics or group dynamics, usability problems that may be discovered by applying these mechanics can be discovered and ironed out during iterative design.

3.1.2 Heuristics for supporting the mechanics of collaboration

I believe that the framework can help inspectors identify usability problems of both groupware prototypes and existing systems. While the framework was developed with low-cost evaluation methods in mind, I had to adapt, restructure, and rephrase it as heuristics, and augment it with a few other important points omitted from the framework. I should emphasize that this was done in cooperation with Gutwin and Greenberg, and there has been a mutual debate and evolution of both my heuristics and how the actual mechanics are being articulated by them over time.

Unlike single-user heuristics that are somewhat independent—chosen by how well they identified 'standard' usability problems (Nielsen 1994b)—the mechanics of collaboration have the advantage that they are linked and interdependent as they collectively describe a partial framework of attributes of how people interact with shared visual workspaces. The resulting eight mechanics of collaboration heuristics are listed in Table 3.1. For each heuristic, I give both the theory that motivates it as well as an explanation of how groupware applications typically realize and support its criteria. Note

that this explanation is somewhat too academic to serve as training material for actual inspectors. Consequently, I include in Appendix A.5 an applied 'training' version of this material oriented towards evaluation practitioners.

1. Provide the means for intentional and appropriate verbal communication The prevalent form of communication between group members is verbal conversations. This establishes a common understanding of the task at hand. Support verbal exchanges or a viable alternative.
2. Provide the mean for intentional and appropriate gestural communication Allow explicit gestures and other visual actions to be visible since they are done in direct support of the conversation and help convey task information. Support illustration, emblem and deixis.
3. Provide consequential communication of an individual's embodiment A person's body interacting with a computational workspace must unintentionally give off information to others. This is the primary mechanism for maintaining awareness and sustaining teamwork. Couple unintentional body language with both the workspace and its artifacts, and the conversation.
4. Provide consequential communication of shared artifacts (i.e. artifact feedthrough) Make artifacts expressive so they give off information as they are manipulated. Support artifact feedthrough.
5. Provide protection Protect users from inadvertently interfering with work that others are doing now, or altering or destroying work that they have done. Provide mechanisms to support social protocols and/or implement technical means to ensure protection.
6. Manage the transitions between tightly and loosely-coupled collaboration Users should be able to focus their attention on different parts of the workspace when performing individual work in order to maintain awareness of others. Provide techniques for making relevant parts of the workspace visible.
7. Support people with the coordination of their actions Support awareness of others' activities to ensure people can coordinate their actions in order to avoid conflicts and make tasks happen in the correct sequence.
8. Facilitate finding collaborators and establishing contact Provide information on potential collaborators so that they can be easily found and their availability for group work can be determined. Initiation of contact should be possible with minimal effort.

Table 3.1 Mechanics of Collaboration heuristics

Heuristic 1: Provide the means for intentional and appropriate verbal communication

Theory. In face-to-face settings, the prevalent form of communication between group members is verbal conversations. These conversations are *intentional* as group members consciously exchange information with each other (Heath et al. 1995, Clark 1996). In turn, this information is typically used to establish a common understanding of the task at hand (Clark 1996). Gutwin (1997) summarizes three ways we pick up information from verbal exchanges:

1. *Direct discussions.* People may explicitly talk about what they are doing and where they are working within a shared workspace. These discussions typically take place when an individual asks a direct question such as "What are you doing?" or when the group is discussing the division of labour.
2. *Overhearing conversations.* People may listen to (i.e., overhear) others' conversations. Although a conversation between two people may not explicitly include a third, people understand that the exchange of information is public and therefore freely available to others.
3. *Verbal shadowing.* People can listen to the running commentary that others tend to produce in conjunction with their actions. That is, people may say what they are doing as they are doing it. This may seem as if people are talking to themselves, but they are really doing it to inform others of their actions. This "verbal shadowing" provides additional information without forcing people to explicitly enter into a conversation. This behaviour has also been called "outlouds" (Heath et al. 1995).

Typical groupware support. For technical reasons, most current visual workspace groupware systems do not support intentional verbal communications directly. Instead, they assume that any communication channel (i.e., text, audio, video) is supplied 'out of band' (e.g., by a telephone link). Depending on the task, an obvious approach to facilitating verbal exchanges within groupware is to have the system itself provide a text chat facility and/or a digital audio link between participants. Each can be implemented in many ways, and each has consequences on how well the communication activity is supported. Text chat, for example, can be via parcel post (e.g., type a line and send) or

real-time (e.g., character by character). Cognoter (Tatar et al. 1991) implemented a parcel-post model of textual communication; items are packaged and sent by the speaker, and then unpackaged and decoded by the receiver. However, if the receiver does not open his "mail" right away, the end result may be multiple messages in no particular order. In addition, people find it cumbersome to carry on long real time discussions through text, and extremely difficult (if not impossible) to type a running commentary as they are performing actions over the visual workspace. Text channels are typically directed at others, so it also may be difficult for a third person to overhear a conversation. While limited, text can be useful for short or sporadic interactions, or where it is impractical to supply an audio link.

In practice, lengthy or highly interactive meetings require an audio channel, typically supplied by a telephone. Digital audio is available in some systems, but currently lacks quality due to reliability, bandwidth, and latency problems. An audio/video channel can also support verbal conversations; however, there are questions concerning the benefits of adding video to an audio channel. Isaac and Tang (1993) suggest that video is useful for managing the mechanics of conversations since visual actions coupled to speech can be seen. Under these circumstances, video can be useful for handling conflict and other interaction-intense activities. Conversely, Egido (1990) found it difficult to engage in negotiation over distributed media since video can introduce distractions that interfere with the accuracy of interpersonal evaluations. Plus, Smith et al. (1989) observed that when conversations shifted to a meta-level, the participants turned away from the monitor displaying the virtual work surface and communicated across the video monitor. Video quality is also notoriously poor in most computers. Images can be blurry and small, and latency can be large. When the frame rate is compromised, images are jumpy and do not really inform others about conversational nuances. While video has appeal and utility, the inspector cannot assume that the inclusion of video as well as audio is the panacea for all intentional communication requirements. A stand-alone audio channel may suffice if:

- the majority of the interaction is performed in conjunction with the shared workspace and its artifacts; and/or

- a high proportion of visual cues are redundantly coded through verbal ones (Short et al. 1976a, Vertegaal 1999).

Heuristic 2: Provide the means for intentional and appropriate gestural communication

Theory. Explicit gestures and other visual actions are also used alongside verbal exchanges to carry out intentional communication. These actions are a crucial resource. Group members use gestures to reference or emphasize workspace objects, express ideas, mediate the group's interactions, focus the attention of others, punctuate talk, and demonstrate actions (Tang and Leifer 1988, Bly 1988). In addition, they communicate information that cannot be readily expressed otherwise. To that end, Tang and Leifer (1988) observed that gestures play a prominent role in all work surface activity for design teams collaborating over paper on tabletops and whiteboards (around 35% of all actions).

These are *intentional* gestures, where people use them to directly support the conversation and communicate task information. Intentional gestural communications can take many forms (Short et al. 1976a, Bly and Minneman 1990, Dix et al. 1993).

1. *Illustration* occurs when speech is acted out or emphasized with gestures. For example, people typically express distances by showing a gap between their thumb and index finger.
2. *Emblem*s occur when gestures replace words such as a nod or shake of the head to indicate 'yes' or 'no'.
3. *Deictic references* or *deixis* happens when people reference objects in the workspace with a combination of intentional gestures and communication (Clark 1996). A common activity is pointing to an object and saying "this one".

Typical groupware support. Because groupware users are separated by distance, gestures remain invisible unless explicitly designed into the system. However, in many groupware applications, the representation of bodies is still completely absent, and gestural communications are lost. Maximizing the information gathered from these activities in a virtual workspace can go a long way in helping people maintain awareness of others and their actions. Groupware must convey and support gestural communication

by making gestures clearly visible, and by maintaining their relation with both workspace objects and voice communications (Tang and Leifer 1988, Tang 1989). In groupware, this is typically done (if at all) via some form of *embodiment.* Common techniques include *telepointers* (Stefik et al. 1987b), *avatars*, and *video images*.

Telepointers are the simplest means for supporting embodiment in a virtual workspace. A person's cursor, made visible to all, allows one to gesture and point to objects in the workspace. While telepointers are limited 2D caricatures of the rich gestures people do with their hands, they are a significant improvement over nothing at all. Early systems, such as GroupSketch (Greenberg and Roseman 1992), were designed to facilitate gestural actions in the following manner:

- each person had their own large and uniquely identifiable telepointer that they could use simultaneously with the others;
- telepointers were always visible within the work surface by all participants;
- they appeared with no apparent delay in order to remain synchronized with verbal exchanges; and
- they maintained their same relative location to the work surface objects across all displays.

Figure 3.1 gives an example of overloading individual telepointers to reflect mode information: Mark's pointer indicates selection mode, Saul's pencil shows that he is drawing, and Dave is trying to grab everyone's attention with the pointing hand.

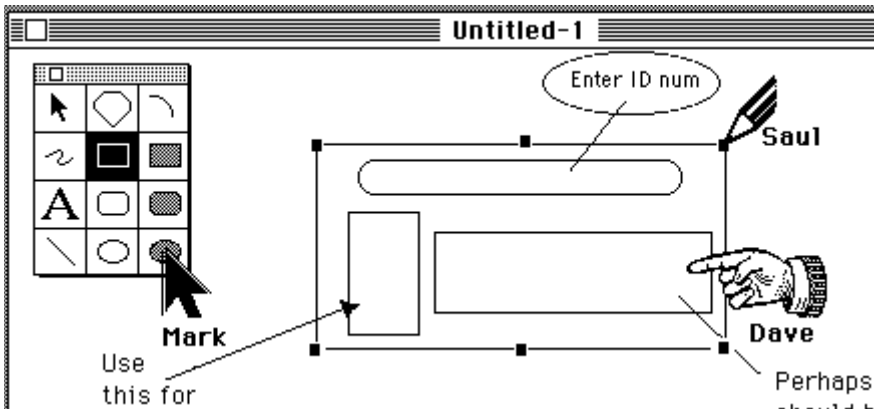

Figure 3.1 Overloaded telepointers (Greenberg et al. 1996)

Avatars are synthetic bodies representing people within a 2D or 3D landscape. While most avatars are extremely crude, some transmit limited hand and body gestures: the idea is to capture real world gestures and recreate them in the simulated space. For instance, TheU from Contact Consortium (Figure 3.2) represents users through avatars for close range chatting. Benford et al. (1995) provides an in-depth exploration on the issues behind the design of user embodiment for collaborative virtual environments (CVEs).

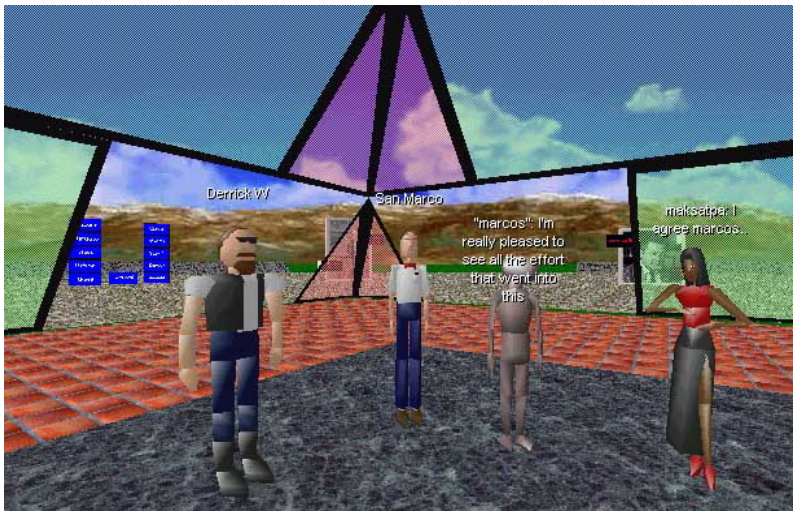

Figure 3.2 Scene from TheU – Virtual University (www.ccon.org)

Finally, *video* can also recreate real bodies within groupware workspaces. Conventional video windows or monitors are insufficient since gestures are detached from workspace objects. A different approach is to mix and fuse the video of the person, their hands, and the worksurface into a single image. When done correctly, the final image comprises both the artifacts and the person, where gestures are maintained relative to the artifacts. For example, VideoDraw (Tang and Minneman 1990) used video cameras mounted above each collaborator's horizontal display to capture (and subsequently transmit) both the drawing marks on the screen and any accompanying hand gestures. As a result, each participant's screen displayed a composite image of the shared workspace consisting of their own marks and the remote participant's video marks and gestures. Similarly, ClearBoard (Ishii et al. 1992, Ishii and Kobayashi 1992) used video cameras

and half-silvered mirrors to combine video of the participant's upper body with the shared workspace (see Figure 3.3). Therefore, collaborators are 'looking through and drawing on a big glass board'—the local user draws on the front of the glass while the remote user appears behind the glass and draws on the back of it.

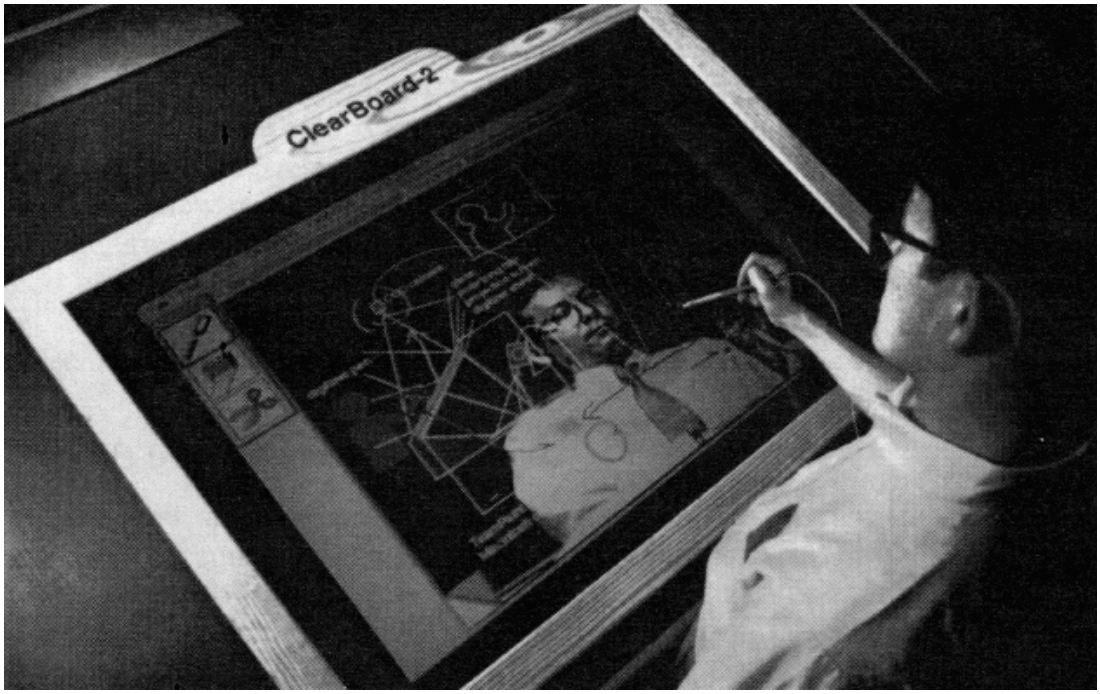

Figure 3.3 ClearBoard-2 in use (Ishii and Kobayashi 1992)

Heuristic 3: Provide consequential communication of an individual's embodiment

Theory. A person's body interacting with a physical workspace is an immediate and continuous source of awareness information. In these settings, bodily actions such as posture, position, contact, and movements of eyes, heads, arms, and hands *unintentionally* give off information, which is picked up by others (Segal 1994). The mechanism of seeing and hearing other individuals active in the shared workspace is called *consequential communication*. Watching other people work is our primary means for understanding what's going on, who else is present in the workspace, their whereabouts, and what they are doing (Gutwin 1997). This flow of information is fundamental for generating and sustaining teamwork (Segal 1994). Consequential bodily

communication is not intentional in the same manner as explicit gestures (heuristic 2). In this case, an individual does not consciously perform the actions to inform others—these are incidental actions that we naturally produce during our interactions. In addition, the perceiver simply gathers what is available from the other.

Unintentional body language can be divided into two categories:

1. *Actions coupled with the workspace* include gaze awareness (i.e., knowing where another person is looking), seeing someone move towards an object, and hearing characteristic sounds as people go about their activities.
2. *Actions coupled to the conversation* are the subtle cues we pick up from our conversational partners that allow us to continually adapt and adjust our verbal conduct (e.g., Sacks et al. 1974, Short et al. 1976a, Goodwin 1981, McLauglin 1984, Clark and Brennan 1991). Cues may be visual, like facial expressions, body language (e.g., head nods), eye contact, and gestures emphasizing talk. Or they may be verbal such as intonation, inflection, pauses, back channels, and the use of particular words. Although these cues may not occur within the workspace, they do provide *conversational awareness* that helps us maintain a sense of what is happening in a conversation. This in turn allows us to mediate turn-taking, focus attention, detect and repair conversational breakdowns, and build a common ground of joint knowledge and activities (Clark 1996). For example, a speaker will use eye contact to assess the listener's attentiveness by starting an utterance, waiting for eye contact from the listener, and then re-starting the utterance (Short et al. 1976a, Goodwin 1981).

Typical groupware support. The key to supporting consequential communication within a real-time shared workspace is to capture and transmit both the explicit and subtle dynamics exchanged between collaborating participants. This is no easy task. The embodiment techniques previously discussed in heuristic 2 are a start. Telepointers, for instance, allow us to see people moving towards an object, which helps to predict their actions. They can also change their shape to reflect a natural action. GroupSketch (Greenberg and Roseman 1992) supported four gesture modes (pointing, writing, erasing, and directing attention) by distinct cursor shapes. For instance, in Figure 3.1 we see that the telepointer is shape of a pencil to indicate Saul is about to draw. Telepointers may

hint at where its owner is looking, although there is no guarantee that the person is really doing so. Avatars can go one step further by linking the 'gaze direction' of the avatar to the point of view. This signals its owner's approximate field of view in the environment. While these techniques are helpful, the impoverish match of these embodiments to a person's actual body movements means that many consequential gestures are not captured or transmitted. Video systems that mix a person's video embodiment into the workspace are more successful at capturing visual cues coupled with the workspace. A notable example is Clearboard (Ishii et al. 1992), which goes to great lengths to keep gaze awareness—whether intentional or consequential—correct (see Figure 3.3). Unfortunately this genre of systems is still quite limited.

CSCW research on conversational awareness has concentrated on how technological mediums affect distributed conversations and how to substitute enough richness in synthetic *video* and *audio* channels to reach an acceptable level of awareness. (e.g., Short et al. 1976a, Egido 1990, Heath and Luff 1991, Fish et al. 1992, Isaacs and Tang 1993)

- *Video.* Special care must be taken with camera placement; otherwise eye contact and gaze awareness will be inaccurate. In most desktop systems, we see speakers 'looking' at our navels or hairline simply because cameras are mounted on top or underneath the monitor. The use of compressed video to preserve bandwidth results in small, jerky and often blurred images that lose many subtle body cues. Even with full video, zooming the camera in to capture facial expressions ('talking head' view) means that other body gestures are not visible. Yet zooming out to include the whole body compromises image fidelity and resolution. To compensate, CoMedi (Coutaz et al. 1999) uses two cameras focused on the same scene, with one giving an 'overview' (say a person at a whiteboard) and the other zoomed in (say the person's face, or where they are writing). Using image techniques, they blend the two together to give a fisheye effect where (say) the person's face is large compared to the rest of the scene. Despite its share of implementation problems, many signals can still be read through a video channel. For instance, facial expressions are still apparent even with poor quality video or small monitors (Dix et al. 1993). A video channel is also useful

for enhancing verbal descriptions, managing extended pauses, and forecasting responses (Isaacs and Tang 1993).

- *Audio.* Audio is also a concern for consequential communication. When the voice channel is of low audio quality, the clarity of a person's speech dynamics is compromised. When the voice is non-directional, people find it difficult to associate a voice with a particular speaker (e.g., multi-point teleconferencing). With half-duplex channels, people cannot speak at the same time, making it harder for listeners to interrupt or to inject back-channel utterances such as 'ok' and 'um'. Also, turn-taking is difficult when more than three individuals participate in a teleconference (Egido 1990). Finally, an audio channel with even small delays can disrupt participants' ability to reach mutual understanding and reduces their satisfaction with the conversation (Isaacs and Tang 1993).

Advanced systems mimic the spatial relationships between people in a multi-point collaboration by letting individuals turn their heads and speak to one another just as they do in real life. This is accomplished by positioning monitors and cameras within and across sites so that all people are seen and heard in the same relative position on their video and audio surrogates. People's images and voices are projected onto separate video monitors and speakers. One early example was the MIT Media Labs' 'talking heads' project (MIT Media Lab 1980). They fashioned a transparent physical mask of a participant, mounted it on a motorized platform at the remote site, and then projected the video image of the participant into the mask. Through sensors, the mask would move to reflect the person's actual head movement.

Heuristic 4: Provide consequential communication of shared artifacts (i.e. artifact feedthrough)

Theory. In face-to-face settings, consequential communication also involves information *unintentionally* given off by artifacts as individuals manipulate them (e.g., Gaver 1991, Dix et al. 1993). This information is called *feedback* when it informs the person who is manipulating the artifact, and *feedthrough* when it informs others who are watching (Dix et al. 1993). Physical artifacts naturally provide visual and acoustic feedback and

feedthrough. Visually, these artifacts depict their state in their physical representation as well as form spatial relationships with one another. In addition, an artifact's appearance sometimes shows traces of its history or wear—gradual and unavoidable change due to use—explaining how it has transformed into its current state (e.g., object wear; Hill et al. 1992). Acoustically, physical artifacts produce characteristic sounds as they are manipulated (e.g., scratch of a pencil on paper) (Gaver 1991, 1993, Gaver et al. 1991). Seeing and/or hearing an artifact as it is being handled helps to determine what others are doing to it and how it has changed.

Another resource available in face-to-face interactions is the ability to identify the person manipulating the artifact. Knowing the action's author provides context for making sense of it, and helps collaborators mediate their interactions. Actions within a shared workspace are often used to bid for turn-taking in a conversation; therefore, being able to associate the action with the initiator helps others yield their turn (Tang 1991).

Due to the spatial separation between artifact and actor, feedthrough tends to be the only vehicle for sharing artifact information amongst groupware participants. However, groupware complicates feedthrough since it limits the "expressivity" of artifacts and actions (Gutwin 1997). For instance, direct manipulation of artifacts in a virtual workspace (e.g., dragging and dropping a small object) is not as visible or distinguishable as its face-to-face equivalents. As a result, these actions can easily go unnoticed and be harder to differentiate from visually similar events. They can also transpire instantaneously (e.g., a click of a button), providing minimal warning of their occurrence and minimal time to see and understand them. In contrast to the physical world, interactions within a virtual workspace are not limited to direct manipulation. Symbolic commands and indirect manipulation afford users the ability to interact with artifacts in ways that are often not possible in the real world. These are shortcuts that stress quick execution while providing minimal feedback. Examples of symbolic manipulation techniques are menu commands, buttons, keyboard shortcuts, and toolbars. In addition to suffering from the same design constraints as direct manipulation, indirect manipulation of a virtual object has little or no workspace representation. It is often difficult to identify the person performing the action even when we can determine the

action's meaning. Unless feedthrough is properly supported by the system, collaboration will be cumbersome.

Typical groupware support. At the lowest level, the shared virtual workspace must display the local user's feedback to all of the remote users. In the event of a direct manipulation of an artifact, the designer must show not only the final position of a moved object but also its selection and the intermediate steps of its move. In groupware, this can be accomplished via *action feedthrough* whereby each individual sees the initial, intermittent, and final state of an artifact as it is manipulated (Gutwin 1997). Early groupware systems imposed "what you see is what I see" (WYSIWIS) view sharing where all participants saw the exact same actions as they occurred in the workspace (e.g., Stefik et al. 1987a). Similarly, feedthrough must be supported during the indirect manipulation of an artifact. *Process feedthrough* ensures that local feedback of a person selecting an operation or command is also transmitted to all others to help them determine what is about to happen (Gutwin 1997). For instance, the local feedback provided by the system when a person hovers over a toolbar button and then press it can be displayed on all screens. Intermediate states of indirect manipulation can also be presented via visual techniques such as action indicators and animation. For example, deleting an object sometimes happens so quickly that others may miss it. To remedy this, we can draw out and highlight the deletion process: we see this in Figure 3.4 where the

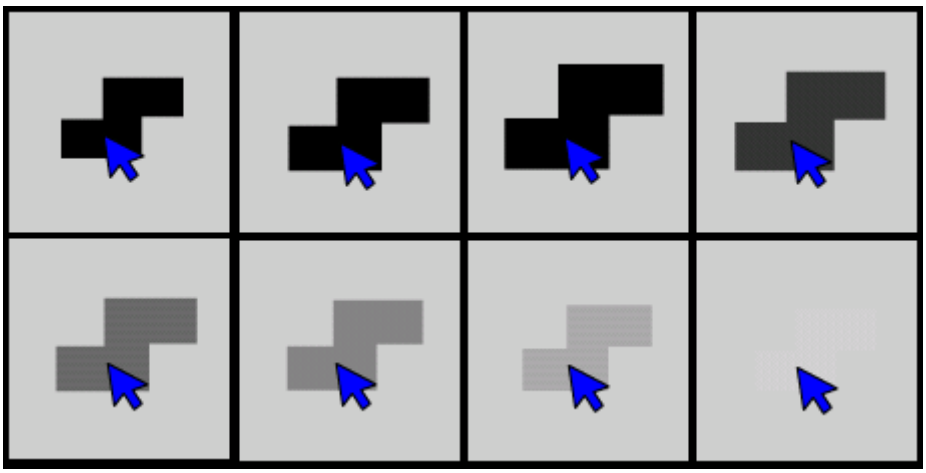

Figure 3.4 Animation to depict the deletion of an object

delete action actually causes the object to grow in size and then slowly fade away. Presenting the artifact's interim feedback to all participants during an operation ensures changes do not happen instantaneously and that information other people can gather about the activity while it is happening is not reduced. In addition, slightly altering the presentation of a remote participant's actions can help distinguish them from their local equivalent (Stefik et al. 1987b).

With both direct and indirect events, all collaborators must be able to identify the action's producer. This helps to provide context to the action. The identity of the individual manipulating the artifacts can be presented via the embodiment techniques previously described in heuristics 2.

Physical objects typically display temporal information regarding how they were created and how they have changed over time through use. In contrast, virtual objects have a consistent appearance and manipulating them does not inherently leave evidence of wear (Hill et al. 1992). Their monotony provides fewer clues for maintaining *change awareness* of the artifact—the ability to track what has happened to the object and what another person has been doing (Tam 2002). To compensate, techniques for displaying the history of virtual artifacts include Hill et al.'s (1992) 'edit wear' and 'read wear' as well as Microsoft Word's 'Track Changes' functionality. Tam (2002) provides a comprehensive study of change awareness theory and techniques for artifacts.

In contrast to physical artifacts, virtual ones also do not naturally produce characteristic sounds; therefore, many groupware workspaces are silent and lacking this form of consequential communication. Sound coupled to an event or action can provide details regarding its occurrence, type, location, and duration even when it happens in a part of the workspace that is not visible. In order to hear sounds in groupware, designers must create and add synthetic replacements to virtual artifacts and events. For example, Gaver (1991, 1993) has been studying auditory icons—the use of everyday sounds to convey information regarding computational activities. Currently, it is difficult determining the best sound to correspond with each action, recreating the subtlety and range of naturally occurring workspace sounds, and finding sounds that work effectively in combination so that each may be heard and understood. In addition, the directional and

proximal components of sounds tend to be weak since workspaces are 2D with limited audio output devices.

Heuristic 5: Provide protection

Theory. In face-to-face settings, physical constraints typically prevent participants from concurrently interacting within a shared workspace. Conversely, groupware enables collaborators to act in parallel within the workspace and simultaneously manipulate shared objects. Concurrent access of this nature can be a valuable resource since collaborators can work in parallel, and because it helps negotiate the use of the shared space. In addition, the competition for conversational turn taking can also be reduced since one person can work in the shared space while another is talking and holding the audio floor (Tang 1991). On the other hand, concurrent access to shared objects can introduce the potential for conflict. People can inadvertently interfere with work that others are doing now, or alter or destroy work that others have done. People should be protected from these situations.

Social scientists have found that people naturally follow *social protocols* for mediating their interactions, such as turn-taking in conversations, and the ways shared physical objects are managed (Clark and Brennan 1991, Tang 1991). People also learn to anticipate each other's actions and take action based on their expectations or predictions of what others are going to do. Therefore, concurrency conflicts may be rare in many groupware sessions—but this can only happen if people have a good sense of what is going on. Therefore, collaborators must be able to keep an eye on their own work, noticing what effects others' actions could have and taking actions to prevent certain kinds of activity. Of course, there are situations where conflict can occur, such as accidental interference due to one person not noticing what another is doing, or unintended interference where one person may not realize that their action may affect another. If conflicts do occur, people are also quite capable of repairing the negative effects of these conflicts and consider it part of their natural dialog (Greenberg et al. 1992). In some (but not all) cases, slight inconsistencies resulting from unresolved conflicts may not be problematic.

Typical groupware support. Groupware users must be able to concurrently interact with the shared virtual workspace (Bly and Minneman 1990). Concurrent access should be allowed to the point of being able to work in the same area at the same time (Tang 1991). In conjunction with this level of concurrent access to the workspace, groupware systems must support protocols to provide protection in an attempt to minimize conflicts (Stefik et al. 1987a, Lauwers and Lantz 1990). Many groupware systems give all collaborators equal rights to all objects. To provide protection, they rely on people's natural abilities to anticipate actions, mediate events, and resolve conflicting interactions. Under these circumstances, the system's role is limited to providing enough awareness of others' actions and feedback of shared objects as well as an appropriate means to communicate with one another. For example, remote handles or a lock can graphically warn users that someone else is already using an item (see Figure 3.5). Although, social protocols will generally work, this approach is not always preferable. For instance, by allowing conflicts to occur, systems force the users to resolve these events after they have been detected. This is undesirable if the result is lost work. Users may prefer 'prevention' to 'cure'. The quality of awareness will also not function well with high-latency communications where there is a delay in delivering one user's actions to others. Finally, social mechanisms are not sufficient for preventing mistakes, conflicting changes, unauthorized access, or malicious actions.

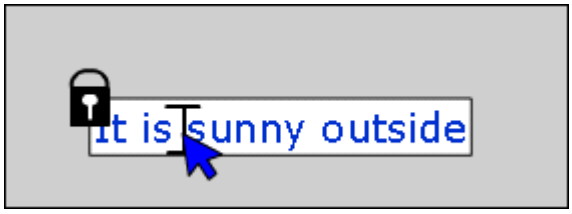

Figure 3.5 Lock to denote that an object is 'in-use'

To assist with social protocols, technical measures such as access control, locking, concurrency control, undo, version control, and turn-taking can be implemented. For example, concurrency control can manage conflicting actions and thus guard against inconsistencies. However, concurrency control in groupware must be handled differently

than traditional database methods since the user is an active part of the process (Ellis and Gibbs 1989, Ellis et al. 1991, Greenberg and Marwood 1994, Munson and Dewan 1996). People performing highly interactive activities may not tolerate delays introduced by conservative locking and serialization schemes. Access control can also regulate access to groupware objects and may be desirable when people wish to have their own private objects that only they can manipulate and/or view. Within groupware, access control must be managed in a lightweight, fine-grained fashion. If not, it will be intrusive: people will fight with the system as they move between available and protected objects.

Heuristic 6: Manage the transitions between tightly and loosely-coupled collaboration

Theory. A shared physical workspace has a dual private/public nature (Tang and Leifer 1988). For instance, individuals can privately work on an idea within a personal portion of the workspace. As the idea matures, it can be made public to everyone sharing a common view of the space (Tang and Leifer 1988, Greenberg et al. 1992). In situations like this, the workspace's dual nature allows individuals to work independently or jointly with others.

Coupling is the extent to which people work together (Salvador et al. 1996). More generally, Gutwin (1997) expresses coupling as "the amount of work that one person can do before they require discussion, instruction, action, information, or consultation with another" (p. 53). People continually shift back and forth between *loosely- and tightly-coupled collaboration* where they move fluidly between individual and group work (e.g., Ishii 1990, Gaver 1991, Dourish and Belotti 1992). To manage these transitions, people need to focus their attention on different parts of the workspace when they are doing individual work in order to maintain awareness of others. Knowing what others are doing or have recently done allows people to recognize when tighter coupling is appropriate. This typically occurs when people need to plan subsequent activities, see an opportunity to collaborate, or have reached a point in their task that necessitates another's contribution. For example, assisting others with their tasks is a vital part of collaboration whereby individuals move from loose to tight coupling. Assistance may be opportunistic

and informal, where the situation makes it easy for one person to help another without a prior request. Awareness of others in these situations helps people determine what assistance is required and what is appropriate. Assistance may also be explicitly requested. Gutwin (1997) observed one participant making an indirect statement requesting assistance, and their partner left their task to help out, and then returned to what they were originally doing. In either case, to assist another, you need to know their current activities, their goals, the stage of their tasks, and the state of their work area.

Typical groupware support. Groupware systems must maintain the dual nature of the workspace so that individuals can choose between private and public work. Depending on the task, each individual must be able to independently navigate the workspace to see and manipulate the objects they need. Implementing traditional WYSIWIS view sharing ensures that people stay aware of one another's activities, but it is often too restrictive when people regularly shift between individual and shared work (Gaver 1991, Dourish and Belotti 1992). Later groupware systems have relaxed the strict WYSIWIS view sharing where people can set their own focal point by independently navigating the shared workspace in order to view objects of interest (e.g., Stefik et al. 1987b). However, when people can look at different areas of the entire workspace, events that occur outside their viewport are not visible. Unless the designer accounts for this, people will lose track of where others are and what they are doing. Maintaining awareness of others when we are not working in the same area of the workspace is further exacerbated because display devices are smaller with lower resolution when compared with the normal human field of view. The reduction in size forces people to work through a small viewport, thus only a small part of a large workspace is visible at any given time. The reduction in resolution makes it harder to see and distinguish artifacts from one another; therefore, visual events can be more difficult to perceive, differentiate, and comprehend. To account for the limitations of these devices, groupware must provide visual or audio techniques that situate awareness in the workspace even when individual viewports vary. Unfortunately, visual techniques encounter the same visibility problem—the relevant part of the workspace has to be visible for the techniques to be useful (Gutwin 1997). If the size of the workspace exceeds a single window, the local user cannot see the entire area

(including objects, events, actions, etc.) outside his or her viewport unless techniques are included to make the relevant parts of the workspace visible. Auditory techniques may help to alleviate this problem since they allow users to be aware of others even when they are neither visible on the screen nor the focus of our visual attention (e.g., Gaver 1991). Visual techniques for situating awareness include *overviews*, *detail views,* and *focus+context views*.

Overviews provide a birds-eye view of the entire workspace in a small secondary window (see Figure 3.6). They correspond to our peripheral vision in a face-to-face setting since they provide a broad perspective without much detail. A properly designed overview makes embodiments, activities, and feedthrough visible, regardless of where they take place in the workspace. As a result, users see both their local view as well as everybody else's whereabouts in the workspace and the general structure of their activities. An overview displaying additional awareness information through telepointers and *view rectangles* (an outline showing what another can see) is called a *radar view*.

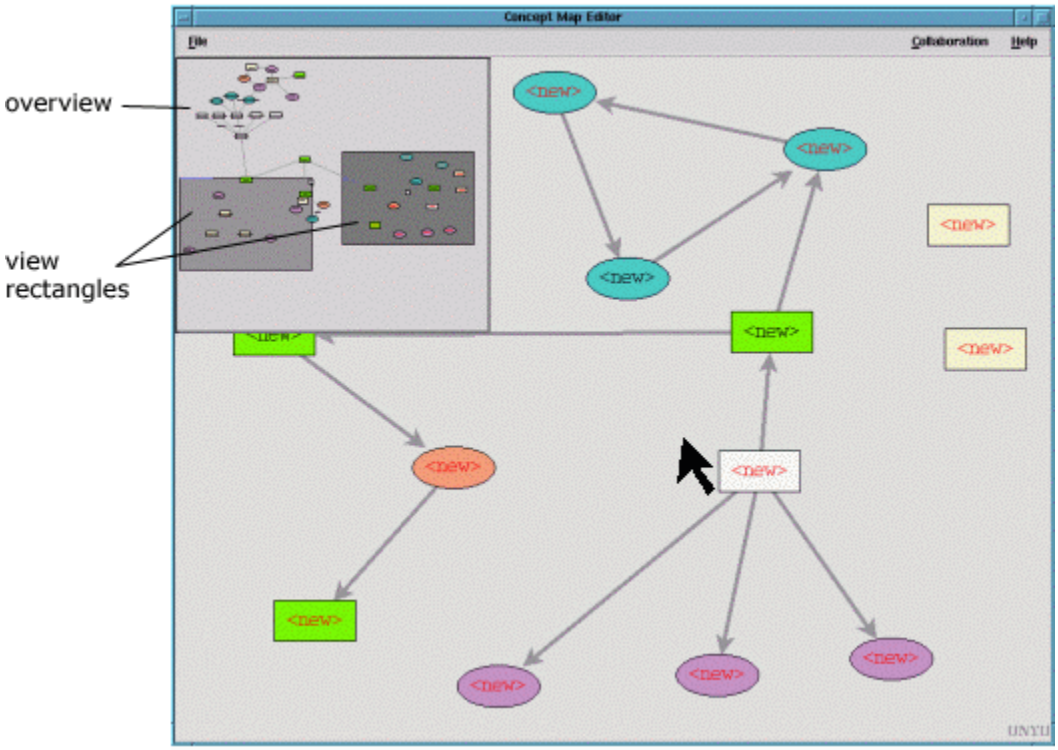

Figure 3.6 Overview example (Gutwin and Greenberg 1998)

With radar views, people can easily pursue individual work by moving their view rectangle in the radar view to a different part of the workspace. Conversely, to work closely together people can quickly align their view rectangle atop one another.

Detail views duplicate a selected part of the shared workspace in a secondary viewport in order to provide a closer look at another person's view of the workspace. In comparison to overviews, they show a smaller portion of the workspace but it is larger with increased resolution (e.g., a cursor's-eye view duplicates the other person's work area and actions in its full size). As a result, one can gather more detailed information regarding the other person's activities.

Focus+context views (Furnas 1986) combine both local detail and global context within a single display through visualization techniques such as:

- Fisheye views: users pick a spot where they wish to see local detail; this area of interest is visually enhanced while the remainder is reduced in size to maintain the overall context (e.g., the hyperbolic browser; Lamping et al. 1995). Figure 3.7 illustrates a concept map editor with a fisheye representation. The left window is the

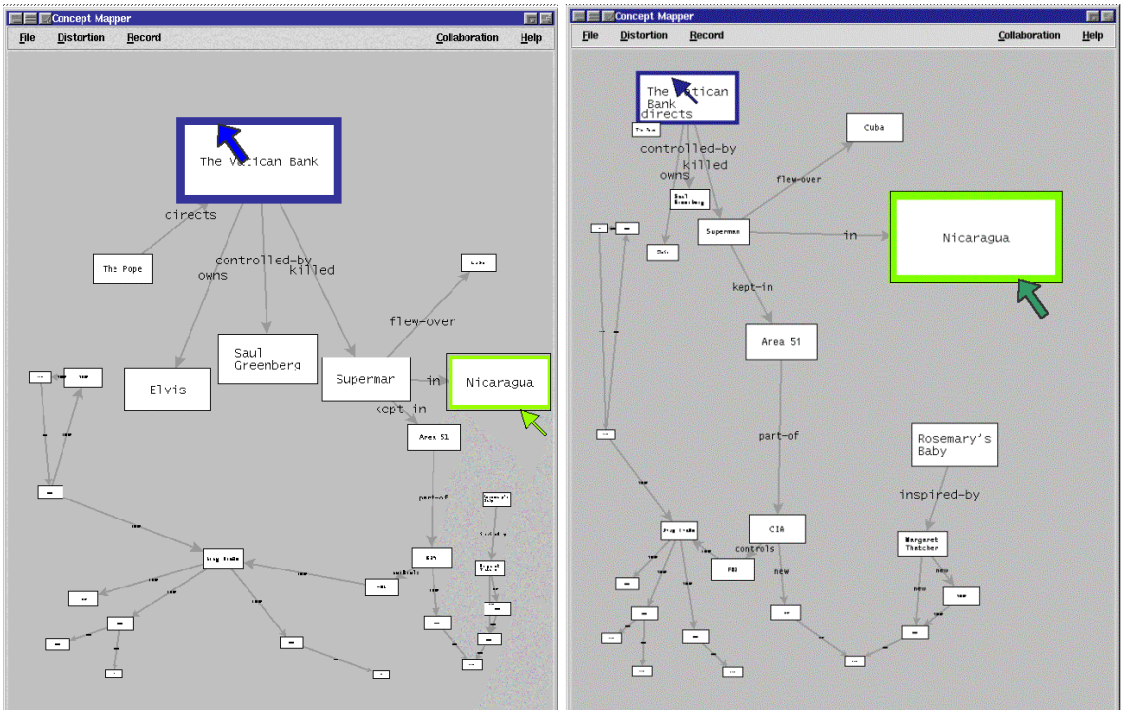

Figure 3.7 Fisheye representation (Gutwin 1997)

local user's view and the right is the remote user's view. The local user's focal point is the "The Vatican Bank" node whereas the remote user's focal point is the "Nicaragua" node. Note the area around the local user's focus is enlarged to support work on these artifacts. The node at each remote user's focal point is also enlarged, but the area around it is not.

- Special lens: dragging a magnifier to the area of the overview where the work is to be done scales the objects beneath it to full size.

Heuristic 7: Support people with the coordination of their actions

Theory. An integral part of face-to-face collaboration is how group members mediate their interaction by taking turns and negotiating the sharing of the common workspace (Tang 1991). People organize their actions in a shared workspace to help avoid conflict with others and efficiently complete the task at hand. Coordinating actions involves making some tasks happen in the right order, at the right time while meeting the task's constraints (Gutwin 1997). The effectiveness of communication and collaboration can be enhanced if the group's activities are coordinated. Symptoms of poor coordination include people bumping into one another, duplication of actions, or multiple individuals attempting to concurrently access shared resources.

Coordination of action is a higher order activity, a necessary overhead when several parties are performing a task (Tang and Leifer 1988, Ellis et al. 1991). It is built upon many mechanisms listed in the previous heuristics. For instance:

- Explicit communication (heuristic 1) is used to accomplish such tasks as discussing how work is to be performed. (Robinson 1991, Fussell et al. 1998).
- Gestures and other forms of visual conduct help coordinate work tasks. Specifically, people use explicit gestures (heuristic 2) to direct attention by referring to existing workspace objects (Tang and Leifer 1988). Similarly, visual cues (heuristic 3) such as facial expressions help mediate conversational turn taking (Heath and Luff 1991).
- Coordination is also accomplished by the way objects are shared during the work process (Robinson 1991). In addition, the way information is generated and organized in the shared workspace acts as a focus that curbs digressions and keeps the group

working in a coordinated fashion (Dourish and Belotti 1992). Both events use workspace awareness (e.g., heuristics 4 and 6) to inform participants about the temporal and spatial boundaries of others' actions. In fact, Dourish and Belotti (1992) state that "awareness information is always required to coordinate group activities, whatever the task domain" (p. 107).

- The workspace itself plays a key role in mediating interactions between collaborators, primarily as a means to focus the group's attention. Collaborators can also occupy their own attention by working privately as well as draw personal attention prior to enacting a gesture. The close physical proximity in these situations helps mediate actions since collaborators are peripherally aware of each other (Tang 1991).

All mechanisms are beneficial for different levels of coordination. At the fine-grained level, for instance, awareness is evident in continuous actions where people are working with shared objects within a confined space. One example is the ability for people to avoid making contact with one another even when collaborating within a small space. On a larger scale, groups regularly reorganize the division of labour (i.e., what each person will do next as the task progresses) through verbal exchanges. These decisions depend, in part, on our awareness of what the other participants are doing, have done, and still have left to do in the task (Gutwin 1997). To that end, knowing activities and whereabouts can help resolve who should do what task in what sequence.

Anticipation also assists the coordination of activities at both levels. At the fine-grained level, people predict events by projecting forward from the immediate past. For example, you might predict that somebody is going to pick up a pen if you see them reaching for it. In turn, you can take action based on this prediction (e.g., pick up the pen and hand it to the other person or alter your own movements to avoid a collision). In this case, anticipation is supported by the up-to-the-moment knowledge of the activity (i.e., where the other person's hand is moving) and the location (i.e., the location of the hand in relation to the pen). In addition, your prediction could have taken into account other information, such as the other person's current activities and if they required a pen. On a larger scale, prediction occurs as people learn which elements of situations and tasks are

repeated and constant. People are extremely adept at recognizing patterns in events, and quickly begin to predict what will happen next in familiar situations (Gutwin 1997).

Typical groupware support. People are generally skilled at coordinating their communication and collaboration with each other (Tang 1989). As a result, groupware should facilitate these abilities and not impose a structure that attempts to manage the interactions (Stefik et al. 1987a, Tang 1991). The visual techniques from heuristics 2 through 5 ensure individuals remain aware of others and their actions within the shared workspace. This workspace awareness provides people with information they need to determine whether current workspace events match previously learned patterns. In addition, implementing relaxed WYSIWIS view sharing (heuristic 6) allows collaborators to see all actions within the context of the entire workspace even when people are working in different parts of it. Finally, collaborators must have the ability to communicate verbally (e.g., via an audio link as suggested in heuristic 1). It is the inclusion of all this support within groupware systems that enables collaborators to effectively coordinate their activities at both a fine-grain level and on a larger scale.

Heuristic 8: Facilitate finding collaborators and establishing contact

Theory. One problem with groupware is that it is not clear how people actually begin their groupware meetings. In everyday life, relatively few meetings are *formal*: scheduled in advance with pre-arranged participants. These are usually arranged via e-mail, telephone, formal meeting requests, etc. In reality most meetings are *informal* encounters: unscheduled, spontaneous or one-person initiated meetings (Lauwers and Lantz 1990). These are facilitated by physical proximity since co-located individuals can effortlessly maintain awareness of who is around. Under these circumstances, people frequently come in contact with one another through casual interactions (e.g., people bump into each other in hallways) and are able to initiate and conduct conversations with little effort. While conversations may not be lengthy, much can occur: people coordinate actions (e.g., "When are you free to discuss the document?"), exchange information, or offer opportunities (e.g., "Come to our presentation."). Successful teams rely on regular, informal, and unplanned contact between their members (Kraut et al. 1988, Cockburn and

Greenberg 1993). It is more difficult to support informal groupware encounters since the bottleneck to rich spontaneous interactions is distance (Kraut et al. 1988).

People are distributed in electronic communities; therefore, designers need to support how the group determines who is around and their availability if they are to initiate contact in a real-time groupware session. Even when potential collaborators have been identified, many mundane factors currently interfere with making contact over computers. People must know electronic addresses and select from many communication channels and applications that are available to the group. For real-time remote conferencing, people must ready software, equipment, and each other well in advance. From a technical perspective, sites may not have the same software; workstations may not support the necessary media (e.g., digital audio); specialized equipment may not be available (e.g., video cameras); poor networks may limit interactions; and applications must run across platforms.

Typical groupware support. Groupware applications must overcome the distance barrier that inhibits informal encounters in order to be successful. Information on potential collaborators must be provided so that they can be easily found and their availability for group work can be determined (Cockburn and Greenberg 1993). If collaborators are able and willing to engage in a groupware session, they must be able to initiate contact with minimal effort (Lauwers and Lantz 1990). Instant messaging provides a simple but limited mechanism for seeing who is around. A richer system supporting both informal awareness and casual interaction is the Notification Collage (see Figure 3.8), where distributed and co-located colleagues comprising a small community post media elements onto a real-time collaborative surface visible to all members (Greenberg and Rounding 2001). People can post live video from desktop cameras, editable sticky notes (which are often used for text conversations), activity indicators, slide shows, desktop snapshots, and web page thumbnails.

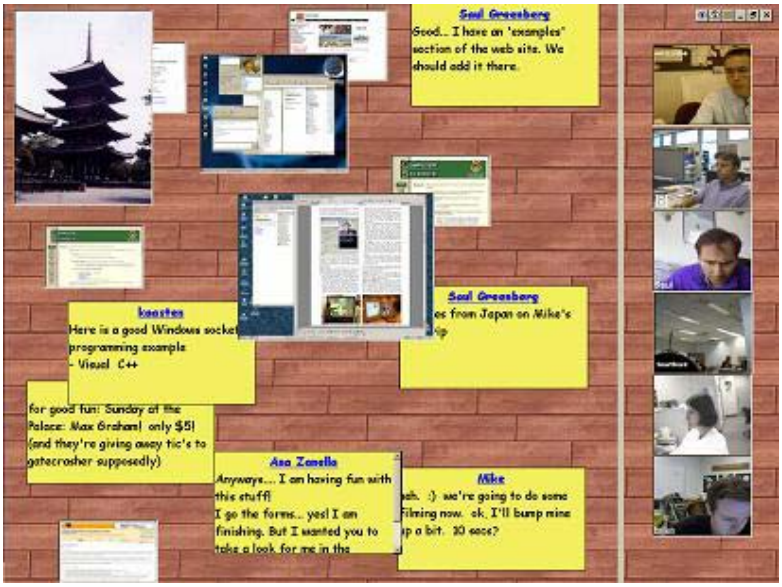

Figure 3.8 Notification Collage

An even more comprehensive approach is TeamWave (Greenberg and Roseman 1998): a room metaphor helps people know who is around and makes it easy to move into conversation and work (Figure 3.9). Specifically, it facilitates the following:

- *Being available.* People can pursue single user activities in a room. Analogous to a physical room used for both individual and group activities, people will be around more often and thus available for real time encounters.
- *Knowing who is around and available for interaction.* Within a spatial setting, we sense who else is around as we walk down hallways, glance into offices, and see others in public spaces. We judge their availability by a variety of cues such as their open door and how busy they look. TeamWave's room metaphor provides a similar sense of presence and awareness by displaying the inhabitants of each room within the electronic community (via a user list), as well as their status information. To judge availability, four 'door states' indicate a person's desire to be interrupted. As in real life, a wide and partially open door icon indicates a willingness to accept interruptions, while a barred door suggests that the room and its inhabitants are inaccessible.

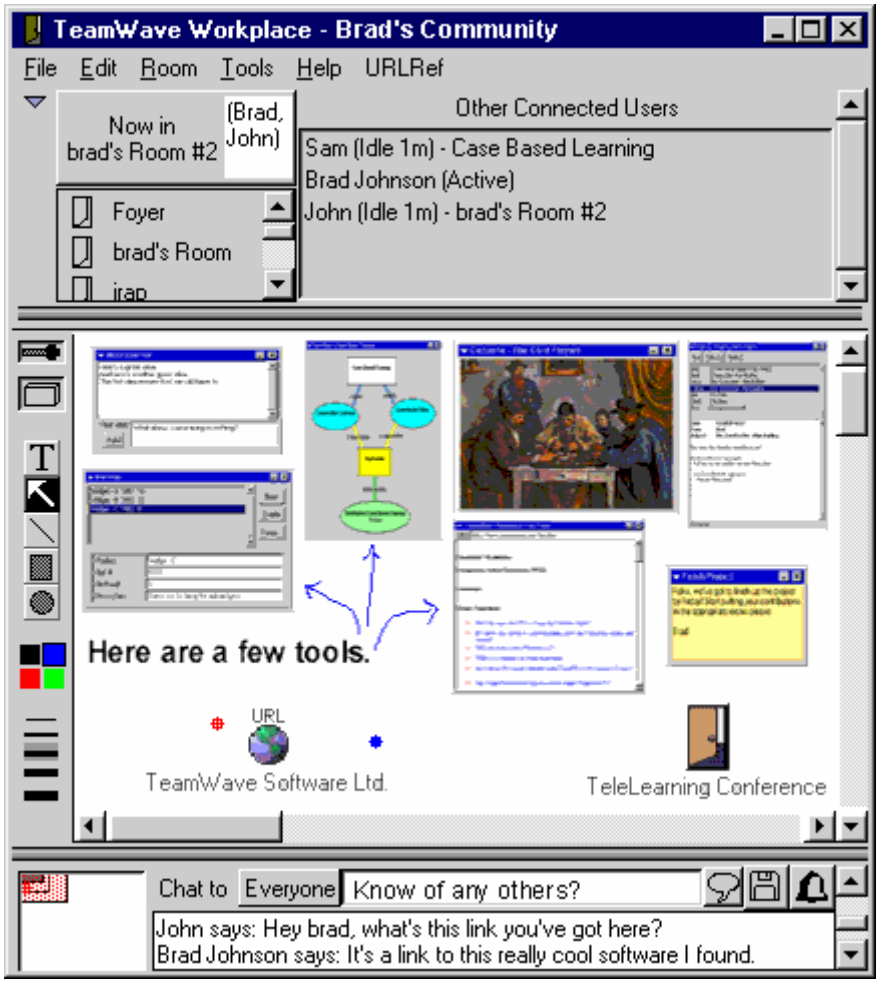

Figure 3.9 A snapshot of a TeamWave room (taken from www.teamwave.com)

- *Establishing contact.* There are several ways to establish contact with others. One can just enter a populated room, and they are immediately connected to others. Room occupants also see the person enter because their picture appears. A room-specific chat facility allows conversations to occur. To initiate a conversation before entering a room, a person can page somebody else. Phone calls can be quickly established since phone numbers are available in each person's business card. To establish contact with others not currently logged on to TeamWave, a person can leave a note in a room suggesting a meeting time and place.

- *Working together.* The power of the room metaphor is that, once in a room, the working context is immediately available. All tools and room artifacts are at hand and new tools can be easily added.

3.1.3 Summary

In summary, these eight heuristics look at the essential mechanical acts of collaboration. The physics of the everyday world as well as people's natural actions means these mechanics 'just happen'. In the computer world, they must be explicitly recognized as well as designed and implemented into the groupware system.

3.2 Locales Framework

3.2.1 Background

The Locales Framework is an approach to help people understand the nature of social activity and teamwork, and how a locale (or place) can support these activities (Fitzpatrick et al. 1996, Fitzpatrick 1998). More formally, the Locales Framework comprises five highly interdependent and overlapping aspects, as summarized below. Further details are in Fitzpatrick et al. (1996) and the framework is described in its entirety in Chapter 8 of Fitzpatrick (1998).

1. *Locale foundations* define a collection of people and artifacts (tools, information, objects) in relation to the central purpose of the social world. A locale within a social world is best considered as a "center" of collective purpose that is part of a dynamic and continually evolving system. Locales are fluid places with social meaning that may be mapped onto physical spaces.
2. *Mutuality* considers those interactions within locales that maintain a sense of shared place. Mutuality includes *presence* information that people and artifacts make available to others, and how people maintain *awareness* of that information. It also includes *capabilities* that entities have to transmit and receive information, and how entities choose from these capabilities to create a particular presence-awareness level.

3. *Individual view over multiple locales* acknowledges that individuals can be participating in many locales. Each person's view is an aggregation of their views onto their locales of interest. People also manifest a "view intensity" onto particular locales as they vary their focus and participation across locales.
4. *Interaction trajectories* concern how courses of action evolve and change over time. In essence, people come into locales and social worlds with past experiences, plans, and actions.
5. *Civic structures* concern how interactions fit within a broader communal level. Civic structures can be considered a "meta-locale" that describe how social worlds and locales relate to one another, how people find their way between them, and how new locales are formed and old ones dissipated.

3.2.2 Heuristics for supporting the Locales Framework

The Locales Framework was not originally developed as a usability evaluation method, rather as a means to understand the nature of social practices in the workaday world and to help inform groupware design. Regardless, it seems amenable to expand this framework into a set of groupware heuristics for several reasons. First, this is a general framework for understanding the fundamental aspects of teamwork. It describes a small set of inter-dependent perspectives of the characteristics of teamwork, where each could be recast as a heuristic. Second, it has been validated as a way to understand existing work practices, and to motivate the design of new systems. Thus it seems reasonable to expect that it can be used as a standard against which to evaluate groupware, particularly those claiming to support long-term team activities within a 'space' or 'place'.

Greenberg et al. (1999) first introduced the Locales Framework heuristics. Based on the text and descriptions found in Chapters 1 and 8 of Fitzpatrick (1998), each of the aforementioned aspects was rewritten as a separate heuristic asking whether a groupware's interface afforded certain social phenomena. Table 3.2 presents a brief summary of these heuristics. I have further expanded each heuristic to render then more understandable for individuals not intimately familiar with the Locales Framework. Since the Locales Framework heuristics are a secondary focus of this research study, I have not included a detailed description of these heuristics in this chapter (as I did with the

mechanics of collaboration heuristics). The expanded version of the heuristics is presented as part of the training material in Appendix A.6.

1. Provide centers (locales) Provide virtual locales as the site, means and resources for a group to pursue team and task work. The system should collect people, artifacts and resources in relation to the central purpose of the social world.
2. Provide awareness (mutuality) within locales People and artifacts must make presence information available to others. People must be aware of others and their activities as they evolve.
3. Allow individual views Individuals should be able to adapt their own idiosyncratic view of the locale or aggregate multiple locales in a way that reflect their responsibilities, activities, and interests.
4. Allow people to manage and stay aware of their evolving interactions over time People must be able to manage and stay aware of their evolving interactions. This involves control over past, present and future aspects of routine and non-routine work.
5. Provide a way to organize and relate locales to one another (civic structures) Locales are rarely independent of one another: people need a way to structure the locales in a meaningful way, to find their way between locales, to create new locales, and to remove old ones.

Table 3.2 Locales Framework heuristics

3.3 Conclusion

In support of my research goal, I have developed two sets heuristics for the purposes of identifying usability problems in groupware. I would not claim that all groupware applications should support all heuristics. Rather, the heuristics can be used to suggest areas of system strengths, and also areas of weakness that might need to be compensated for in other ways.

While these heuristics are tentative, I believe they are good candidates. Unlike Nielsen's heuristics, each set is derived from a well-defined theory or framework of group work. Thus they not only include a specific scope or context, but they should have good coverage (if the theories are in fact complete). In the next chapter, I outline the

methodology used to validate the heuristics' ability to account for groupware usability problems.

Chapter 4: Evaluation Methodology

Having formulated two sets of groupware heuristics from two inter-related frameworks in Chapter 3, the next logical task in my research is to validate these heuristics. To that extent, this chapter describes the two-step methodology used to carry out this objective. The first step was a pilot study whereby the groupware heuristics were reviewed and subsequently modified prior to conducting the main research study, the second step. The main research study was set up to mirror the methodology and terminology employed by Nielsen to validate his heuristics (Nielsen and Molich 1990, Nielsen 1992). For our purposes, two groups of inspectors with varying degrees of expertise in HCI and CSCW evaluated two groupware systems, a toy groupware editor called GroupDraw (Roseman and Greenberg 1994), and a very substantial commercial groupware system called Groove (www.groove.net). Their goal was to record as many usability problems as possible that violated the groupware heuristics. In Chapter 5, I will analyze these identified problems to derive conclusions regarding the practicality of the heuristics.

In this chapter I set the foundation for the evaluation by first re-stating my objective in accordance with my pre-defined research goals (Section 4.1). Next, I present the pilot study used to validate the training material and the changes to the heuristics in response to the findings (Section 4.2). Finally, I describe the methodology used to carry out the main research study as well as the rationale behind some choices (Section 4.3).

4.1 Objective

As per my research goals (see Section 1.4), the objective of validating the heuristics is to:

> "Demonstrate that the adapted heuristic evaluation for groupware remains a 'discount' usability technique by analyzing the ability of inspectors to identify problems in collaborative applications".

As a means to execute the stated objective, I revisit Nielsen's motivations for traditional heuristic evaluation where he designed it as a usability engineering methodology that can

be done *cheaply, quickly*, and produce *useful results* (Nielsen and Molich 1990, Mack and Nielsen 1994). To briefly elaborate the three key terms:

1. *Cheaply*. Nielsen's heuristic evaluation places a low demand on resources. Non-experts can carry out inspections; therefore, it is not confined to using more costly experts (the caveat: experts will produce better results) (Nielsen 1992). This is practical because heuristic evaluation is, in practice, relatively easy to learn and to apply (Mack and Nielsen 1994). Consequently, extensive resources are not required for training. Finally, heuristic evaluation requires only the evaluator, the interface, paper, and pencil. No special equipment or facilities are necessary.
2. *Quickly.* Heuristic evaluations do not require advance planning nor do they require a large amount of time to conduct (Nielsen and Molich 1990). Typically, the evaluation of an interface can be done within an hour or two for a simple interface and somewhat longer for more complex interfaces.
3. *Useful results.* Despite performing heuristic evaluations with less control and formality than would be entailed by formal user testing, this technique provably produces useful results (Nielsen and Molich 1990, Mack and Nielsen 1994). As discussed in Chapter 2, only a few evaluators (3-5) can discover a majority (~75%) of interface bugs with varying severity. Fixing these bugs can in turn improve the usability of products.

Within my main research study, I look to validate heuristics applied to groupware evaluation to see if it remains a discount usability technique that is quick and cheap to perform while producing useful results. To gauge this cost-effectiveness, I conducted the evaluation in the following manner:

- I chose as inspectors people with knowledge of human computer interaction, but limited knowledge of Computer Supported Cooperative Work.
- Prior to performing their evaluations, inspectors received a basic one hour training lecture and a packet of written materials (see Appendix A) explaining the heuristics.
- Inspectors self-selected the length of time and amount of effort they would put into completing the evaluation.

Given the first two conditions, if non-groupware specialists are able to effectively use the heuristics to uncover usability problems after only modest training, then the technique requires minimal human resources. In other words, the technique would not necessitate inspectors with special groupware skills who are expensive and hard to find. If basic training suffices, then extensive training costs would not be incurred. The third condition looks to replicate real world practices for using this methodology. In the end, we expect (as with traditional heuristic evaluation) that a modest aggregate of 3-5 inspectors can uncover an acceptable proportion of a system's problems (~75% of known usability problems). With groupware heuristic evaluation, we hope for a similar range if this is to be an effective but cheap discount usability evaluation technique.

Despite our desire to achieve the same advantages as traditional heuristic evaluation, there are some inherent shortcomings with using this technique to assess groupware. One problem with groupware evaluation is that you typically need another person to help test the system's capabilities. This affects both the cheapness and quickness of the technique and introduces more planning. In addition, this may adversely affect the variety of usability problems discovered by evaluators during their inspection. I will discuss this issue in more detail later. Plus, groupware often involves more time to set-up, especially if it requires special hardware such as microphones, headsets, and video cameras.

4.2 Pilot study

The groupware heuristics used by the inspectors to perform their evaluation of the two systems are presented in Appendix A.5 and A.6. A quick glance over the mechanics of collaboration heuristics reveals that their explanation and format differ from the same heuristics presented in the previous chapter. Originally, my approach was to provide the inspectors with the mechanics heuristics as formatted in Chapter 3 for their evaluations. I assumed that this would have been adequate for the main research study. To gauge the accuracy of this assumption, I administered an informal review of the mechanics of

collaboration heuristics prior to the official evaluations taking place. This is the crux of the pilot study.

Due to time constraints and my secondary focus on the Locales Framework heuristics, the pilot study was conducted with only the mechanics of collaboration heuristics. However, the findings from the study were also used to improve the Locales Framework heuristics.

4.2.1 Participants

Three professional HCI and groupware practitioners with 5 to 17 years of relevant experience were recruited to review the mechanics of collaboration heuristics. All three were also currently engaged in a project that involved re-designing the user interface for a real-time, shared workspace collaborative application.

4.2.2 Method

Each participant received a copy of the mechanics of collaboration heuristics similar to what is found in Chapter 3. That is, it was a somewhat academic description of the heuristics (what they received was actually an earlier draft of that Chapter). While reading the heuristics, each was asked to address the following two questions:

1. Do I understand the principles of the heuristic?
2. Would I be able to apply this heuristic as part of a heuristic evaluation?

The objective was to gain informal feedback regarding the comprehensibility of each heuristic and its suitability to be applied in the context of an evaluation. Feedback was gathered in the form of written comments on the original handouts and verbal comments recorded from interviews with each individual after their review.

4.2.3 Results

Overall, the reviewers' feedback was positive regarding the content of each mechanics of collaboration heuristic. They expressed that beginning a heuristic with its supporting theory—derived from studies of face-to-face interactions—helped to establish its motivation. In addition, the reviewers considered the heuristics to be practical since they included examples of techniques employed by groupware systems to comply with each one. This feedback reflected my original motivations for structuring the heuristics in this

format. Despite the positive feedback, two main areas of concerns surfaced in response to the reviewers answering the two aforementioned questions.

The first area of concern surrounds the heuristics' comprehension. Each reviewer had to re-read each heuristic (in some cases, several times) in order to comfortably understand the concepts. Amongst other things, this lead to some confusion regarding the break down of information between two heuristics. In one instance, one reviewer did not fully understand the difference between intentional and consequential gestural communication (heuristics 2 & 3). Although this difference is somewhat arbitrary and sometimes blurry, there is a distinction that must be clearly spelt out to ensure the effectiveness of the heuristics.

The second area of concern centered on the ability of the reviewers to apply the heuristics as part of a groupware evaluation. The reviewers expressed difficulties extracting the pertinent facts from the heuristic since each one did not explicitly state what one should look for when inspecting a system. One reviewer was expecting to see statements clearly identifying the compliant criteria such as a 'system should provide …'. Plus, a number of the heuristics list problems with existing techniques when it comes to supporting the theory as opposed to stating how to apply the concepts. Bundling up these comments, the reviewers concluded that it would be awkward to use the heuristics in their current format to effectively evaluate groupware systems.

4.2.4 Discussion

Prior to conducting the main research study, the mechanics of collaboration heuristics (and subsequently the Locales Framework heuristics) had to be revised to address the reviewers' concerns. I did not want the bottleneck to 'good' results to be the inability of the inspectors to understand and apply the heuristics. This risk had to be minimized.

To address the first concern, all of the heuristics were re-written with the intent of making them an 'easier read'. Domain specific terms were replaced with more common terms. Sentences were shortened. In addition, all new concepts introduced by the heuristics were clearly defined and spelt out.

The second concern regarding the practicality of the heuristics in their current format raised an interesting issue, one that I had not considered up until this point. It is

one thing to have all the pertinent theory encapsulated in a set of heuristics, but it is another to ensure that the heuristics are packaged as a practitioner's training tool to facilitate conducting a heuristic evaluation. With the latter, an evaluator should be able to easily extract the necessary information from each heuristic in order to identify usability problems with a system. The naïve approach is to structure these heuristics as a checklist whereby a practitioner is presented with a series of items that can be systematically checked off during an inspection. However, this defeats the strength of the heuristics: a small set of motherhood statements that act as a guide to uncover a broad range of usability design principles. In addition, each heuristic may need to be interpreted differently depending on the context of its use. A checklist may not lend itself to this flexibility. Consequently, the heuristics require a limited amount of judgment on the part of the inspector to be effectively applied.

In response to the reviewers' comments, the mechanics of collaboration heuristics were re-structured in an attempt to facilitate their role as a tool. The first section of each heuristic was divided into two new sections "Theory" and "What this means for groupware". "Theory" provides the underlying principles behind each heuristic and is not critical to performing an evaluation. The intent of the next section, "What this means for groupware", is to provide all the pertinent information that an inspector should consult when performing a heuristic evaluation. The final section "Techniques used in groupware" remained essentially intact with the "Typical groupware support" section from the early version of the heuristics.

4.2.5 Sanity check

To ensure that all comments had been adequately addressed, the same three professionals reviewed the revised mechanics of collaboration heuristics. All were comfortable that their comments had been addressed. They viewed the new heuristics as easier to read and understand. This set of heuristics and its training material is presented in Appendix A.5.

However, one reviewer still raised some concerns regarding the ability for practitioners to use the information as presented as a tool. This particular reviewer suggested that each heuristic be broken down into finite requirements with a scale whereby one could rate the ability of each system to comply to the requirement. This gets

back to the argument between developing a set of guidelines versus motherhood statements. Our goal was in-line with Nielsen's standpoint that a set of motherhood statements is more beneficial, hence the decision was made to continue with the research study with the heuristics in this format.

4.2.6 Locales Framework heuristics

Although the pilot study was conducted with the mechanics of collaboration heuristics, the findings were transferable to the Locales Framework heuristics. Consequently, the latter were re-written to address the issues in a similar manner to the mechanics of collaboration heuristics. The only difference between the two sets of heuristics is the lack of a "Techniques used in groupware" section with the locales heuristics. Unlike the mechanics, there is no clear set of ways to support a locale; indeed, this remains an on-going effort in research. The scattered inclusion of the few tentative techniques now available may bias the reader to just look for those. As a result, the section was omitted from the heuristics.

4.3 Main research study

Subsequent to revising both sets of heuristics in response to the pilot study findings, my next step was to see if these adapted heuristics could be used for "heuristic evaluation of groupware and remain a 'discount' usability technique". To do this, I analyze the ability of inspectors to identify problems in two collaborative applications. The next sections outline the methodology implemented during the main research study. This includes providing details regarding the participants, materials, method, and data collection.

4.3.1 Participants

To assess the practicality of the groupware heuristics, I looked at the ability of individuals with minimal training in HCI but with varying levels of knowledge in CSCW to apply them. We recruited several groups fitting these criteria and validated our demographics through a questionnaire (see Background Information questionnaire in

Appendix A.2) that asked them to assess their knowledge and experience in HCI and CSCW. Participants were categorized as two evaluator types: *novice* and *regular*.

- *Novice evaluators* were 16 students in their 3rd or 4th year of a University computer science program. All had completed one full course in HCI and were currently enrolled in a second senior-level advanced undergraduate HCI course. When asked, the majority indicated some experience with designing and evaluating graphical user interfaces. However, few had any substantive knowledge regarding CSCW interface design principles. Consequently, the group consisted of "novices" with respect to CSCW usability but not with respect to computers and HCI.
- *Regular specialists* were 2 professors and 9 students working on their graduate degrees in computer science. All had a history of research, applied work, and/or class work in groupware and CSCW, as well as conventional user interface design and evaluation. These individuals were labeled 'regular specialists' since in contrast to the former group; they were knowledgeable of groupware fundamentals.

We must stress that except for the professors, all participants were students. Due to their limited availability, professional HCI/CSCW practitioners from industry were not employed as part of our research. I speculate that this category of inspectors would perform better than the regular specialists given their expertise and practical experience in the field of HCI and CSCW. Therefore, if the two categories of inspectors chosen for our research can effectively apply the groupware heuristics then a logical assumption is that professional practitioners could do at least as well.

4.3.2 Materials

To help inspectors conduct their heuristic evaluation, we gave them a *training packet*, *workstations*, and the two *groupware systems*.

Training packet. The training packet (Appendix A) consisted of two sets of groupware heuristics; one based on the mechanics of collaboration (A.5) and the other on the Locales Framework (A.6). These were revised versions in accordance with the pilot study findings. The packet also contains the following forms:

- a consent form outlining the inspectors' participation in the research study (A.1);

- a background information questionnaire to help ascertain each inspector's knowledge and experience in the areas of HCI and CSCW (A.2);
- pre- and post-training feedback forms containing questions assessing the ease with which the inspectors were able to comprehend the heuristics before and after their training session (A.4); and
- problem reports designed in accordance with Nielsen (1994a) and Cox (1998) and used by the inspectors to capture their usability problems (A.7).

Workstations. The novice evaluators conducted the heuristic evaluations of the two groupware systems on four PC workstations located in a single row in the undergraduate computer lab. For the purposes of the research study, a graduate student installed the systems on these workstations. The regular specialists installed the two systems on their personal workstations located in their respective labs. Installation instructions (Appendix A.3) were provided on how to set-up the software.

Groupware systems. As part of the study, the participants evaluated two quite different shared visual workspaces contained in two real-time groupware systems: *GroupDraw* and *Groove*.

GroupDraw is an object-oriented 'toy' drawing program built to show people how to program the GroupKit groupware toolkit (Roseman and Greenberg 1996). It includes basic features found in most structured drawing packages. A 'GroupDraw conference' is launched from the Conferences menu item found in an 'Open Registration' window (Figure 4.1). This window includes a list of all on-going conferences that the user is currently participating in (e.g., GroupDraw, Hello World), the members in a selected conference (e.g., Kevin, Bob, and Susan are participating in a GroupDraw conference), and the ability to join an existing conference by double-clicking on it.

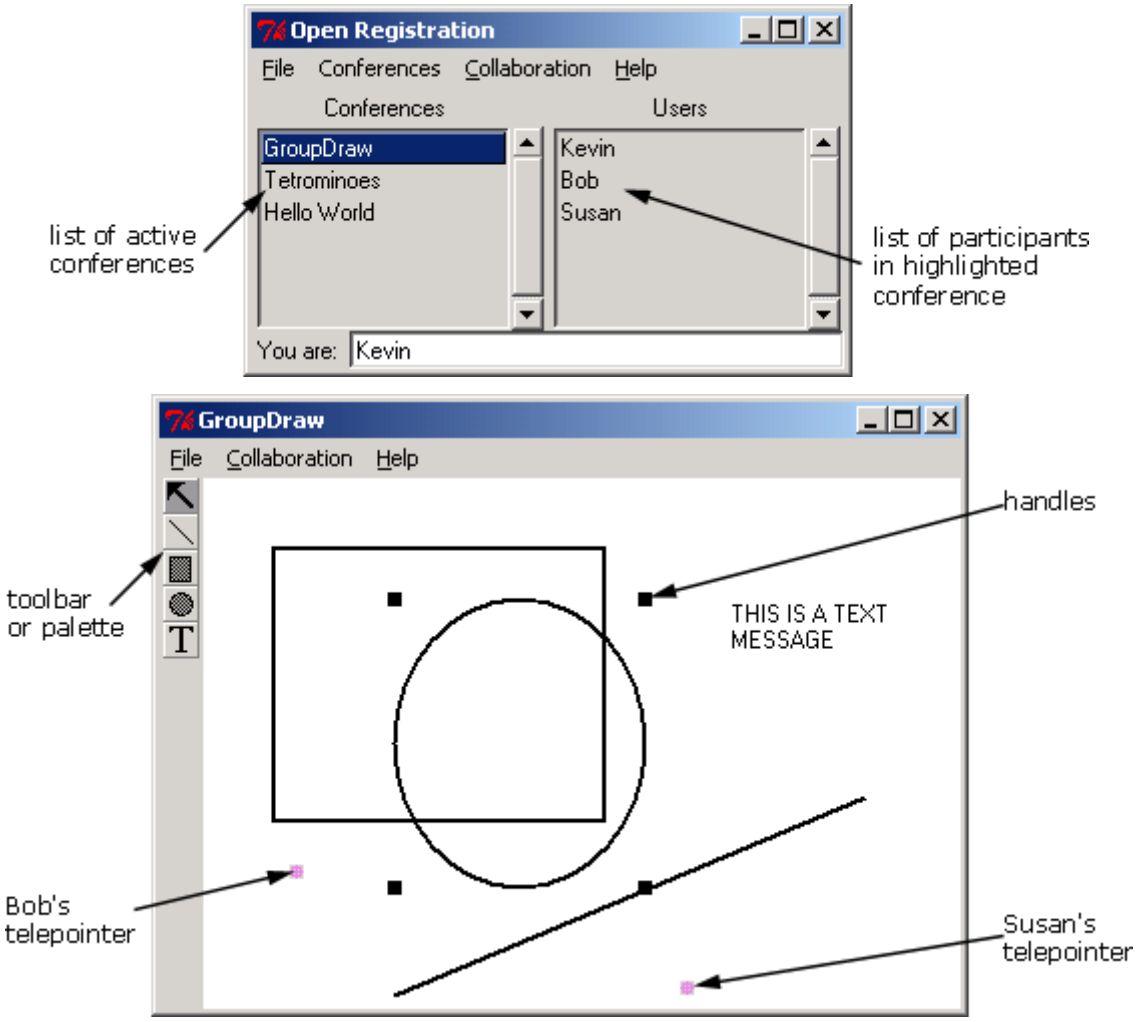

Figure 4.1 Open Registration dialog box and GroupDraw shared workspace

After joining the GroupDraw conference, the shared workspace appears. Figure 4.1 (bottom) illustrates GroupDraw with three people (Kevin, Bob, and Susan) working on a rudimentary drawing. GroupDraw supports multiple active cursors, displayed as small cross-hairs. As visible in the screen snapshot, users can work simultaneously where they can create, move, resize, and delete drawing objects (rectangles, lines, circles, text, etc). Participants can communicate with one another by sending text notes to each other through the notes subsystem (Figure 4.2).

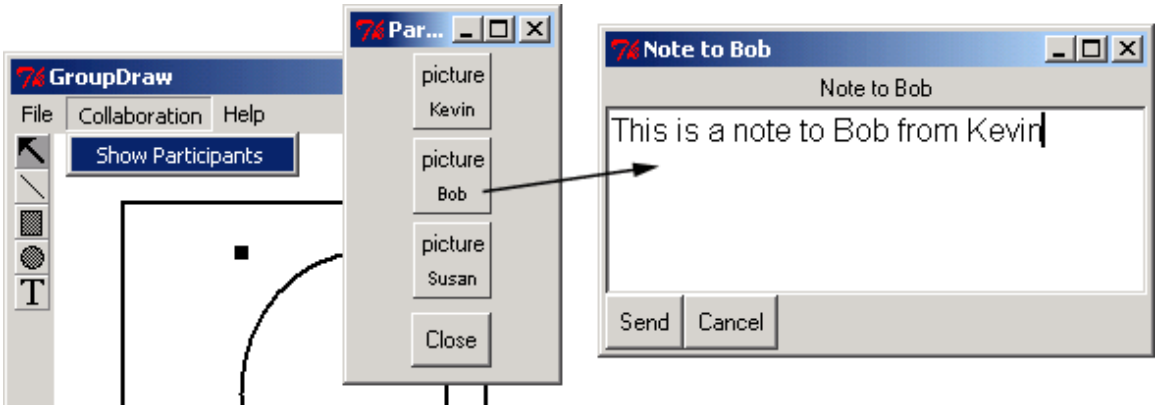

Figure 4.2 GroupDraw's Notes functionality

Because GroupDraw was developed in our laboratory, we were familiar with its functionality and had some a priori knowledge of its problems.

Groove is a virtual space for real-time, small group interactions. Users create "shared spaces" to communicate and collaborate with one another. Changes made to a shared space by one participant are automatically synchronized with all other computers. Groove is far more comprehensive than GroupDraw: it is best viewed as an environment where many tools are available. Its functionality includes:

1. Communication tools – live voice over the Internet, instant messaging, text-based chat, and threaded discussion.
2. Content sharing tools – shared files, pictures, and contacts.
3. Joint activity tools – co-Web browsing, multiple-user drawing and editing, group calendar.

Figure 4.3 is a snapshot of a shared workspace called 'Groove Space'. The left side lists the space's two members and their presence (Kevin is present in the workspace whereas Saul is not logged on to Groove). Currently the Outliner tool is active in the shared space. Distributed collaborators use this tool to brainstorm and hierarchically organize ideas in real-time. Participants can switch tools by selecting a tab (e.g., Sketchpad, Notepad). Real-time communication among the members can be done via Groove's voice or text chat facility in the bottom portion of the screen.

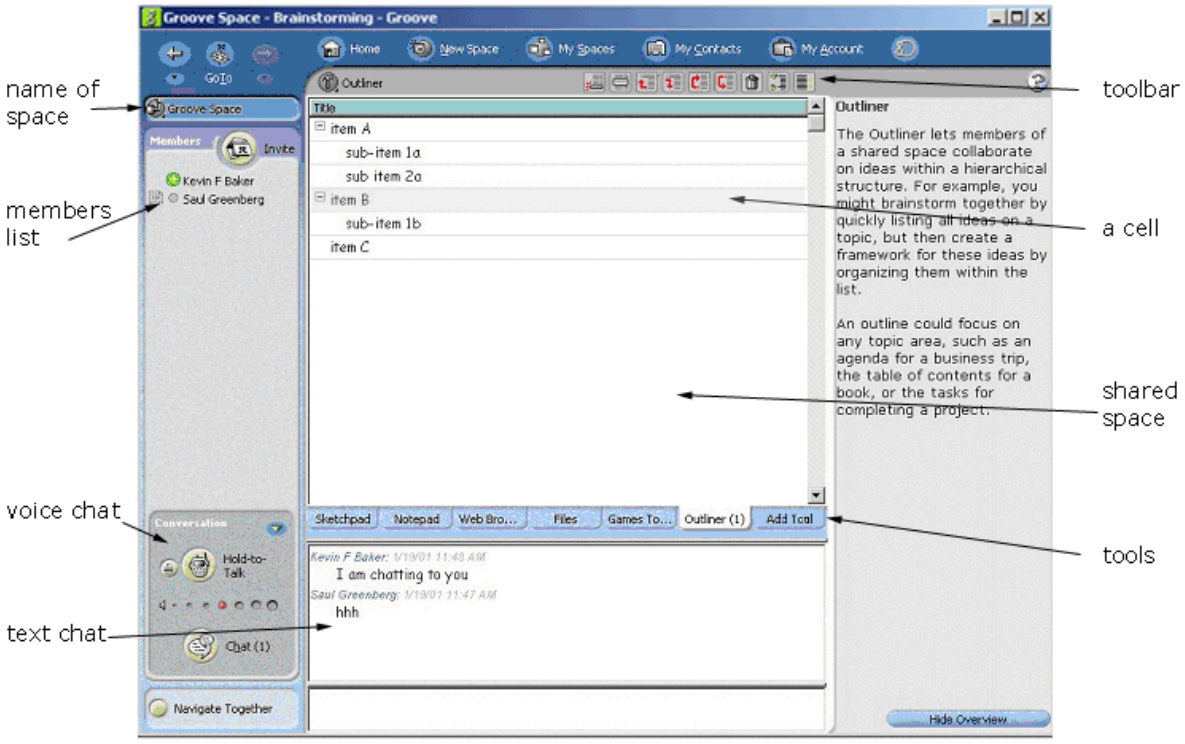

Figure 4.3 Groove Outliner tool

Groove was chosen as a system to evaluate for this study since it is commercially available professional product.[1] We had no prior experience with it and had no preconceived notion of how effective it would be as groupware.

4.3.3 Method

The heuristic evaluation of the GroupDraw and Groove interfaces followed Nielsen's standard recommendations (refer to chapter 2 for details). This involved administering an orientation session to each group of inspectors prior to the evaluation process. However, I did not conduct a debriefing session due to the geographical separation between the inspectors and the researcher (myself). The following section elaborates the *orientation session* and *evaluation process*.

[1] We used a Groove 'pre-release', and I suspect it has changed considerably since then. The subsequent evaluation results only concerns the version available at the time of this study, and thus I make no claim on the usability of today's Groove product.

Orientation session. In preparation for the orientation session, I distributed all the training materials and forms to the participants for their review ahead of time. Prior to the session, each participant was asked to:

- sign the consent form (A.1);
- fill-out the background information questionnaire (A.2);
- read the detailed written description of the groupware heuristics (A.5 and A.6); and
- complete the pre-training feedback form (A.4).

Given the number of inspectors (27 total inspectors) and their location (22 in Calgary and 5 in Saskatoon), I conducted three separate 90-minute orientation sessions to three audiences. Each session subscribed to the same format. At the beginning, I collected the signed consent form as well as the completed background questionnaire and pre-training feedback form from each inspector. In return, the inspectors were handed a blank post-training feedback form (this form was identical to the pre-training feedback form) and an ample supply of blank problem reports for their heuristic evaluation of the systems.

Next, I conducted a one-hour training session on the proposed groupware heuristics. This included a review of the theory supporting each heuristics, how to apply them during an evaluation, and real-time groupware examples that illustrated compliance and non-compliance with each heuristic.[2] To that extent, the goal of the training was to ensure that everybody had a good understanding of the principles behind each heuristic and how to use them. The informal nature of the session also provided the forum for individuals to clear up any misconceptions or uncertainties. The participants then filled out a post-training feedback form so that I could gauge how well they comprehended the groupware heuristics upon receiving the training. The completed feedback forms were collected at the close of the orientation session.

Subsequent to the training on the heuristics, I provided the inspectors with an overview of the two groupware systems under test, GroupDraw and Groove. First, I

[2] The primary focus of the research was to assess the mechanics of collaboration heuristics; consequently everybody was trained on this set of heuristics. Only the 11 regular specialists were trained on the Locales Framework heuristics.

highlighted the pertinent features of each system. Since both systems contain many features and tools, I then instructed the inspectors on the functionality they should inspect during the heuristic evaluation. For GroupDraw, the inspectors were asked to evaluate only the shared workspace (Figure 4.1 bottom) and the Notes functionality (Figure 4.2) as the means for communicating with one another. For Groove, I asked the inspectors to evaluate the Outliner tool (Figure 4.3) as well as the text chat and audio link.

Finally, general instructions were given regarding the process for conducting the heuristic evaluation. The novices were to inspect both systems with only the mechanics of collaboration heuristics. The regular specialists were to assess the groupware interfaces with both the mechanics of collaboration and Locales Framework heuristics. Since there is partial overlap between the two sets of heuristics, the regular specialists were instructed to record only those problems with the Locales Framework heuristic that were not previously reported using the mechanics of collaboration heuristics.

In addition to recording usability problems, I instructed the inspectors to perform the heuristic evaluation through self-guided exploration. The lack of pre-prepared scenarios assesses the utility of the heuristics without investing a significant amount of preparation time. This further translates to evaluating the strength of the heuristic evaluation as a stand-alone technique in the 'worst' case scenario. Under these circumstances, the inspectors were not provided any assistance with their evaluation. In addition, both systems are not complex in their design since they are intended for the general population as a walk-up-and-use interface. Consequently, scenarios are not required to supplement a lack of domain knowledge on the user's part.

It was imperative for the inspectors to perform their evaluations independently of one another. A section of the research is reserved to determining if aggregated evaluation produce better results than the individual evaluations. Consequently, if the evaluators performed group inspections, the expectation is that the variety of usability problems uncovered will be smaller. The variety is the reason for the improvement one gets when using aggregates of evaluators (Nielsen and Molich 1990). As mentioned earlier, given the inherent nature of groupware it is difficult for inspectors to perform an evaluation truly independent of one another. To effectively evaluate the systems' teamwork

dynamics, individuals are forced to at least work in pairs. Within this context, the inspectors might tend to bias each other towards a certain way of approaching the inspection and therefore limit the usability problems they uncover. This is especially true when two inspectors encounter a problem and subsequently discuss the problem in an attempt to understand what has occurred. While having another person at the other end of the groupware system is helpful, I asked the inspectors to minimize their discussions about the problems they saw.

Evaluation process. Other than receiving the general instructions for conducting the evaluation process during the orientation session, the inspectors had complete freedom regarding their approach and performance of the heuristic evaluation (which reflects real world practices). This is similar to Nielsen's evaluations of traditional heuristic evaluation.

Each inspector dictated when, where, and how to perform the evaluation. As with traditional heuristic evaluation, they could use the heuristics to systematically review all the functionality. Alternatively, they could walk through an imaginary task of their own making and verify how each step of the task complied with the heuristics. Inspectors had complete control over the length of time and amount of effort they for completing the evaluation. In some instances, the evaluation was performed in pairs and other times it involved four or five inspectors working concurrently.

4.3.4 Data collection.

For each problem uncovered, the inspectors completed a separate problem report by recording a description of the problem, the violated heuristic, a severity rating, and an (optional) solution to the problem. They judged a 'major' severity rating as one that represented a significant obstacle to effective collaboration, while a 'minor' rating is one that could be worked around by the participant. A blank problem report is found in Appendix A.7.

With respect to formulating severity ratings for each usability problem, Nielsen (1994a) states that evaluators have difficulty performing this step during the evaluation process since they are more focused on finding new usability problems. In addition, each

evaluator will not find all the usability problems in the system; therefore, the severity ratings will be incomplete since they only reflect those problems found by the evaluator. Despite these arguments, the inspectors were asked to rate the severity of each usability problem they uncovered since this was quick and was not considered a hindrance to their evaluation. Plus, it provides insight into each inspector's assessment of the problem's impact with using the system. These original problem reports form the raw data for my analysis in the next chapter (see Appendix B for all problem reports).

4.4 Conclusion

This chapter reviewed the methodology for the pilot study and the subsequent changes to the groupware heuristics in preparation for the main study. Next, the main research study was introduced via a detailed description of its methodology. Of primary importance is that we used people that we felt were reasonable approximations of the actual practitioners we would expect to do groupware evaluations, and that our methodology echoed Nielsen's traditional heuristic evaluation methodology.

In Chapter 5, I compliment this chapter by presenting the analysis and results of the main research study.

Chapter 5: Analysis and Results

Chapter 4 provided an introduction to the main research study by describing the methodology used for validating the two sets of groupware heuristics. As a follow-up, this chapter analyzes the raw data produced by the 27 evaluators. Specifically, I examine their relative performance and variability identifying the known teamwork usability problems in two collaborative applications in order to derive conclusions regarding the practicality of the heuristics. To that end, I employ an analysis similar in many ways to Nielsen's approach to validating his traditional heuristic evaluation methodology (Nielsen and Molich 1990, Nielsen 1992).

To set the scene, this chapter first walks through a method called *results synthesis,* a process that I used to distil the inspectors' raw problem reports into a consolidated list of known teamwork usability problems for each groupware system (Section 5.1). These lists form the data for my subsequent analysis of the heuristics. Next, I present the results from my analysis and my interpretation of their meaning (Sections 5.2 and 5.3). Finally, I reflect upon the experiment and some of the factors affecting the results and how I interpret them (Section 5.4). Similar to Chapter 3, the analysis is split into two separate parts; one dealing with the mechanics of collaboration heuristics and another with the Locales Framework heuristics.

5.1 Results synthesis

Individual inspectors produced numerous problem reports as a means to record the usability problems they uncovered during their evaluations of the systems with the groupware heuristics. These reports cannot be compared directly with each other because inspectors often identify the same problem with different terminology and/or different levels of abstraction. Thus I needed to analyze and transform the raw problem reports into a form where they could be directly comparable and meaningful to analyze. Using a heuristic evaluation method called *results synthesis* (Cox 1998), I distilled the large number of raw problem reports collected from the individual inspectors into a concise

aggregated list of known teamwork usability problems for GroupDraw and Groove (Appendix B). I now outline my method and the subsequent findings.

5.1.1 Method

Step 1. To ensure that the raw data is captured in a suitable fashion, I gave the inspectors blank problem reports to record their usability problems (see section 4.3.5 for details). As an example, Figure 5.1 illustrates a problem report filled out by one inspector during his review of the GroupDraw interface with the mechanics of collaboration heuristics.

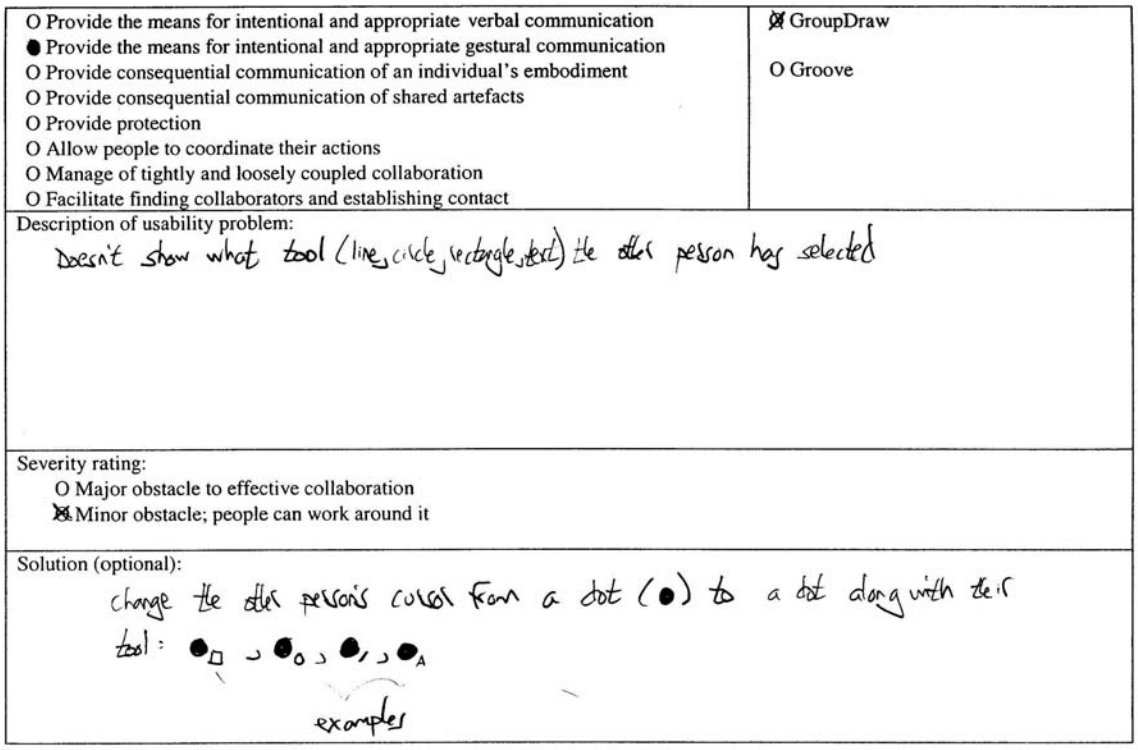

Figure 5.1 Sample of a completed problem report

In this case, the inspector specified that the problem violated only one heuristic. However, on many occasions inspectors indicated that a given problem violated multiple heuristics. Typically, the description of the usability problem is brief (one or two sentences). In roughly half of the reports, the inspectors provided an optional solution to the problem.

Step 2. Next, I transcribed verbatim all the problem reports (only the problem descriptions and solutions) into an Excel spreadsheet and initially categorized them according to their author. In addition, if an inspector listed multiple problems in a single problem report, I separated them into one problem per report. This raised an initial issue in regards to the granularity with which to define a single teamwork usability problem. For instance, one inspector stated that the GroupDraw telepointer did not point on the tool palette nor did it point on the menu. These two issues are tightly coupled and by all means could be labeled as one problem: the telepointer does not point outside the shared space. However, I chose to separate them into two distinct problems: 1) the telepointer does not point on the tool palette and 2) the telepointer does not point on the menu. The reason was that although some inspectors did uncover one problem, they did not necessarily capture the other. One could argue that from a practical standpoint, trying to fix one problem will more than likely lead to fixing the other one. Regardless, the decision was to record problems in their finest level of granularity as opposed to lumping them together under an 'umbrella' problem.

Step 3. I then classified each problem according to one of the following categories:

1. *Raw teamwork problem:* a problem that can be categorized according to one of the groupware heuristics. These are used to derive the consolidated list of known teamwork usability problems for both systems.
2. *Out of scope:* a problem that is not categorized according to the groupware heuristics.
3. *False positive:* a 'problem' that turned out not to be a usability problem at all.

Step 4. I then grouped all of the raw teamwork problems (category 1) according to the eight mechanics of collaboration and five Locales Framework heuristics. This categorization scheme was chosen to accommodate the forthcoming analysis. However, depending on the desired outcome of a heuristic evaluation this may not always be the optimal course of action. Cox's (1998) participants, for instance, defined most of their final groupings into the 'best' way for dealing with the interface problems as opposed to the violated heuristics. Within my research study, I did not always classify each problem

according to the heuristic originally recorded by the inspectors in their problem reports. Often different inspectors saw the same problem from divergent perspectives and therefore categorized the problem with different heuristics. In these situations, I chose the heuristic that seemed the best match for the duplicate problems.

While rearranging the raw problems according to the heuristics, I also grouped together the most obvious duplicates and treated them as a single entity.

Step 5. At this point, the initial problem groupings from the previous step underwent further refinements as I now dealt with the more ambiguous issues surrounding their synthesis. This activity is particularly time consuming and onerous, since individual inspectors describe problems in their own unique way. It is not always immediately apparent that multiple inspectors are in fact addressing the same problem. Inspectors may use dissimilar terminology for the same situation. They may also identify different symptoms for the same fundamental problem (i.e. root cause) or describe them at quite different levels of abstraction. Two separate inspectors evaluating Groove, for instance, raised these two teamwork problems that stem from the system's complete lack of compliance with one of the mechanics of collaboration heuristics:

1. "The system does not support consequential communication of an individual's embodiment."
2. "I don't have a clue where somebody is within the shared space and what they are doing."

At a high level, both problems address the same issue; however, the level of abstraction is different. The first describes the problem's root cause; the system does not support embodiment within the shared workspace. The second problem provides the symptoms or consequences for not supporting embodiment; it is hard to determine where others are in the tool and what they are doing during the collaboration. Even though both problems are not described at identical levels of abstraction, they are obviously related and therefore I grouped them together as a single teamwork usability problem.

Step 6. Now, I described each grouping as a single teamwork usability problem in terms of the symptoms as opposed to the root cause. This level of abstraction was chosen since

the purpose of this research is to compile a list of known teamwork usability problems for each system and not solutions to the problems.

Step 7. Finally, I rated each teamwork usability problem as major versus minor based on the ratings and argumentation of the inspectors. The final consolidated list of known teamwork problems with severity ratings for both interfaces are reported in Appendix B.

Given the constraints in the research study, I independently performed the results synthesis to produce the final teamwork usability problems with severity ratings for both systems. These circumstances naturally raise concerns regarding the reliability of the results since judgment is required to determine:

- whether the reported problem is a teamwork usability problem and not 'out of scope' or a 'false positive';
- the categorization of teamwork problems within heuristics;
- the consolidation of duplicate problems; and
- the severity of the final teamwork problems.

Relying on a single evaluator to produce these results can introduce bias and unreliability. For example, severity ratings gathered and averaged from several evaluators is likely more reliable than a severity rating from a single individual. Nielsen (1994a) recommends ratings from three to four evaluators for many practical purposes. To reduce this bias I had another usability practitioner independently review and confirm my results. While not a rigorous review, that evaluator felt that the final teamwork usability problems were reasonable. Only a modest number of items were changed.

5.1.2 Results

Table 5.1 summarizes the breakdown of the inspectors' original problem reports into the four results synthesis categories (see Appendix B for complete list of problems). The 'raw teamwork problems', 'out of scope', and 'false positives' are the results at the end of Step 3—duplicates are not removed. The 'total known teamwork problems' are the consolidated problems resulting from the entire results synthesis process. In other words, the 321 raw teamwork problems for GroupDraw and 331 for Groove (uncovered using the mechanics heuristics) were distilled into 62 unique and distinct problems for

GroupDraw and 43 for Groove. Similarly, GroupDraw's 28 and Groove's 66 raw teamwork problems (uncovered using the locales heuristics) are distilled into 15 and 28 distinct problems respectively. As a cautionary note, these problems are not necessarily the complete set of teamwork usability problems for GroupDraw and Groove since it is impossible to guarantee that the inspectors reported every single problem.

Heuristics	System	Raw teamwork problems	Out of scope	False positive	Total known teamwork problems
Mechanics of Collaboration	GroupDraw	321	15	14	62
	Groove	331	21	15	43
Locales Framework	GroupDraw	31	0	1	15
	Groove	66	0	1	28

Table 5.1 Breakdown of original problem reports after result synthesis

5.1.3 Discussion

From Table 5.1, we see that inspectors recorded a significant number of problem reports with the mechanics of collaboration heuristics for GroupDraw and Groove—over three hundred for each. The result is a long list of teamwork problems for both systems. From this initial analysis, we can conclude that these heuristics are definitely helpful in uncovering bugs related to the support of real-time collaboration within a shared workspace. Conversely, the inspectors produced significantly fewer problem reports (and subsequently fewer teamwork problems) when inspecting both systems with the Locales Framework heuristics. Even though fewer inspectors performed this evaluation, this raises initial questions concerning the practicality of these heuristics. In both cases, the relatively small number of 'Out of Scope' problems and 'False Positives' indicate that inspectors stayed focus on collecting groupware usability problems.

With the mechanics of collaboration heuristics, the ratio of initial raw teamwork problems to the final list of total known teamwork problems is ~6:1 for both GroupDraw and Groove. This indicates a significant amount of redundancy in the problems by the inspectors, which is in keeping with traditional outcomes of results synthesis (Cox 1998).

On the other hand, there is little redundancy with the problems reported by the inspectors with the locales heuristics (ratio equals ~2:1)

5.2 Mechanics of collaboration – Analysis and interpretation

Through the results synthesis analysis, I can conclude that a large group of inspectors will identify numerous teamwork usability problems with the mechanics of collaboration heuristics. In addition, there is a significant amount of redundancy in the reported problems. Therefore, can we rely on a small number of evaluators to find the majority of problems? To answer this question, the usability problems captured using mechanics of collaboration heuristics and subsequently assimilated through results synthesis are analyzed against the individual and aggregate performance of the inspectors. Specifically, the analysis is broken down into three sub-sections:

1. *Relative performance of individual inspectors* analyzes the average performance of individual inspectors, evaluates each inspector's ability to find specific problems, and compares the results across the two inspector types and the two groupware systems.
2. *Performance of aggregates of inspectors* studies the total number of problems found by aggregates of inspectors for each system.
3. *Characterizing found usability problems* analyses the inspectors' performance based on categorizing problems according to their severity and the heuristics violated.

This roughly replicates Nielsen's methodology and analysis (Nielsen and Molich 1990, Nielsen 1992), which allows me to compare the results brought forth by this research study with his traditional heuristic evaluation. If the results are similar, then I can claim that the groupware heuristics also achieve a 'discount' status comparable to traditional heuristic evaluation for single-user systems.

For the purposes of the analysis, inspectors were scored for the usability problems they reported during the heuristic evaluations of GroupDraw and Groove. The scoring was done by matching each inspector's set of problem reports against the final compiled list of known teamwork usability problems. Credit was given for mentioning a usability problem even if it was not described completely.

5.2.1 Relative performance of individual inspectors

In this section, I analyze the performance of individual inspectors using the groupware heuristics to uncover known usability problems. In addition, I compare the performance across the two types of inspectors and across the two systems.

5.2.1.1 Average performance of individual inspectors

The most basic form of evaluation is to look at the average number of problems found by each inspector. This helps to determine if we can rely on a single individual to uncover a significant proportion of the total known usability problems.

Results. Table 5.2 breaks down the average performance of individual evaluators according to the groupware systems (GroupDraw *vs.* Groove) and the inspector type (novice *vs.* regular). The table also provides the amalgamated results across the inspector types and systems. Figure 5.2 graphs this data as it compares the proportion of inspectors and the proportion of the total number of usability problems they found. As a comparison, I overlay data on traditional heuristic evaluation collected from 77 inspectors of the Mantel system and 34 inspectors of the Savings system (Nielsen and Molich 1990).

System	Evaluators	Total known teamwork problems	Best evaluator (%)	Worst evaluator (%)	Average problems found (%)	Standard deviation
GroupDraw	16 novice	62	33.9	8.1	20.3	7.5
	9 regular*	62	40.3	1.6	14.0	12.3
	All	62	40.3	1.6	18.0	10.0
Groove	16 novice	43	53.5	16.3	27.9	11.0
	11 regular	43	46.5	9.3	22.2	9.5
	All	43	53.5	9.3	25.6	10.8
Both	All novice	105	41.0	11.8	24.1	10.3
	All regular	105	43.9	8.6	18.5	11.9
	All	105	43.9	8.6	21.9	10.2

*Two regular specialists did not perform a heuristic evaluation of the GroupDraw interface.

Table 5.2 Individual differences in evaluator's ability to find usability problems

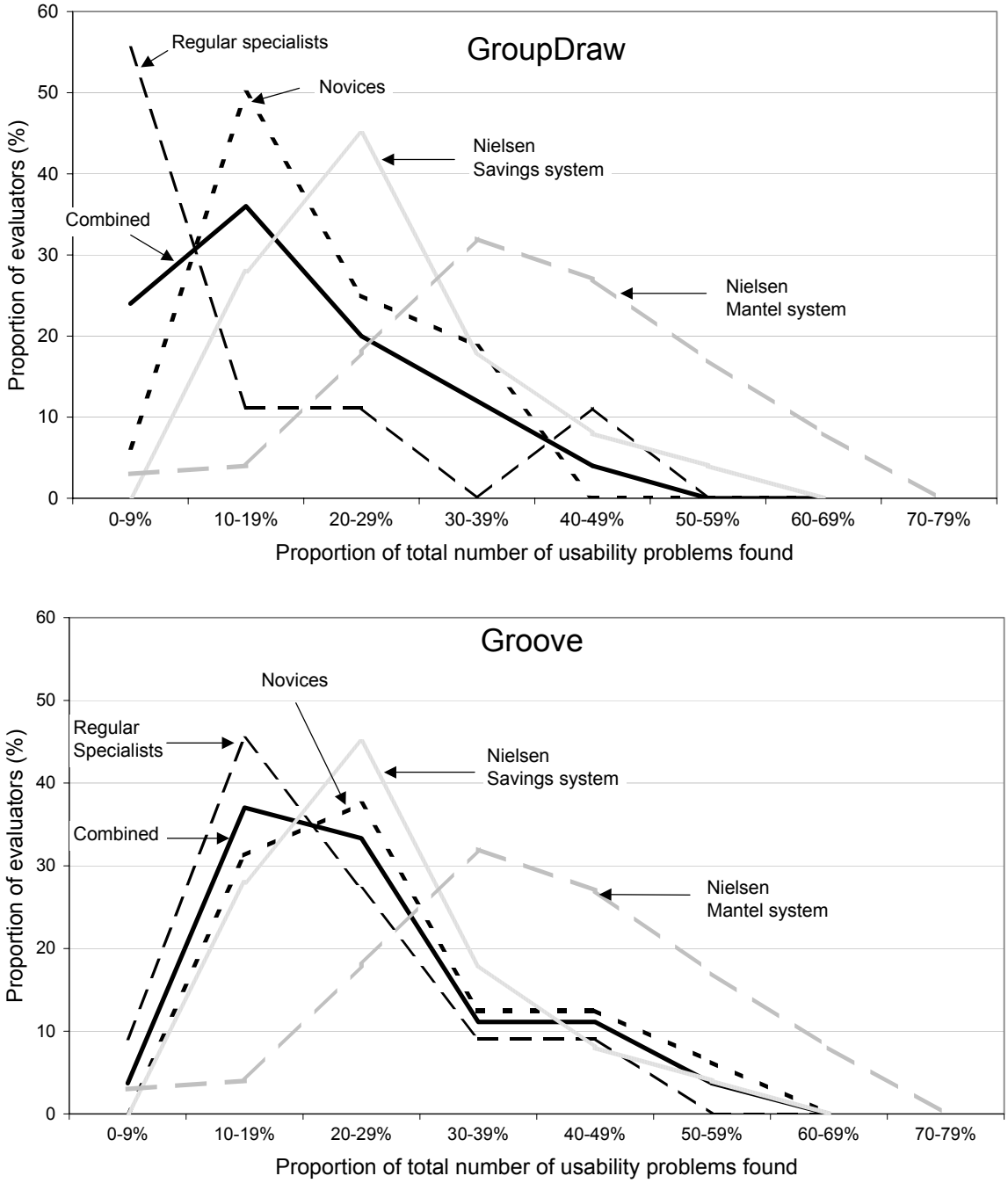

Figure 5.2 Distributions of the number of usability problems found by the evaluators

With both systems, no single evaluator found all the known usability problems. On average, the number of problems found by each category of inspectors is relatively

low. Across both systems, novices found ~24% of the problems whereas the regular specialists found ~19%. As a comparison, Nielsen's (1992) novice evaluators uncovered 22% of known usability problems in one system and the regular specialists uncovered on average 35% of problems across six systems (ranging from 20% to 51%).

The table and graphs illustrate a significant difference between individual performances. For both systems, the best novice evaluator uncovered 41% of the known usability problems; whereas the worst found only ~12%. Similarly, the best regular specialist found ~44% of the problems and ~9% for the worst. In comparison to the traditional heuristic evaluation of four systems, the best regular specialist found between ~47% and ~75% of the problems and the worse discovered between 0% and ~23% (Nielsen and Molich 1990). Nielsen does not provide these results for novice evaluators. Figure 5.2 provides a clearer picture of the actual (roughly normal) performance distribution—the majority of the evaluators do about average with only a few doing quite well and a few doing quite poorly. Overall, the groupware inspectors perform well although their performance is on the lower end of Nielsen's evaluators for traditional heuristic evaluation (as seen by the overlaid data on Figure 5.2).

Discussion. Given the large number of problems uncovered during the heuristic evaluation of each interface, it is unrealistic to assume that one evaluator would discover all problems. However, the findings from this study are more or less in line with those found in traditional heuristic evaluation, where one evaluator uncovers (on average) only a modest number of problems compared to the total found problems (Nielsen 1992). The results also show that inspector performance with the groupware heuristics is somewhat less than inspectors doing traditional heuristic evaluation. Possible reasons for this lesser performance include:

1. As a continuation of a concern first brought forth by the reviewers in the pilot study (section 4.2.4), inspectors may have found the current version of the groupware heuristics more difficult to learn and apply when compared to normal heuristics.
2. Inspectors (even the regular specialists) did not have the same amount of exposure and personal experience with groupware as compared to traditional inspectors' exposure to single-user systems.

3. Post-evaluation feedback from several individuals expressed a difficulty evaluating Groove's Outliner tool since it was 'so bad'. The tool fails to comply with many of the heuristics at the lowest level. This made it difficult to understand and evaluate the system's confusing behaviour.
4. The 1-hour training session for the groupware heuristics and evaluation methodology may have been insufficient.

At this point, I can only speculate on the reasons. Further studies would have to be conducted to help narrow done the true cause.

According to t-tests, Nielsen (1992) determined that the difference in performance between evaluators with varying levels of expertise was statistically significant. Specifically, usability specialists are better than people without usability training, and double specialists are better than usability specialists due to their knowledge with the specific type of user interface under test. My expectations prior to the groupware heuristic evaluations was that the regular specialists would discover, on average, more usability problems than the novice evaluators due to their increased expertise in CSCW. However, this hypothesis was incorrect as the regular specialists, who all had good background in groupware, performed more poorly than the novices. To understand this outcome, I interviewed several regular specialists regarding their participation in the research study. Their feedback suggests two reasons for their poorer performance: reduced incentive and the large time commitment. Additionally, I suspect a third factor, the geographical separation between the test conductor and participants, may have had an adverse effect on their level of effort.

First, all evaluators were participating in the research study on a purely voluntary basis. Evaluations had to be completed on their own time, above and beyond current obligations. The amount of time invested in the heuristic evaluations was self-chosen and there was no monetary remuneration. For the regular specialists, their only incentive to complete the heuristic evaluations was their willingness to assist another graduate student. In contrast, the novice evaluators performed the evaluations as a requirement for a graded course and thus had a high incentive to do the task diligently. Consequently, the

regular specialists were less motivated than the novices and therefore put forth less effort to completing the task.

Second, the increased time commitment imposed on the regular specialists also influenced their level of effort. The novices only evaluated each system with the mechanics of collaboration heuristics. In addition to this task, the regular specialists inspected the systems with the Locales Framework heuristics. Post-evaluation discussions with one regular specialist indicated that evaluating two systems with both sets of heuristics took much longer than expected. Time was required to understand the heuristics, set-up and familiarize oneself with the systems, conduct the heuristic evaluations, and record usability problems. As a result, some of the regular specialists said they only spent a modest amount of time performing the actual heuristic evaluations.

Third, the geographical separation between the test participants in Calgary and Saskatoon and the primary test conductor in Ottawa may have been a negative factor. Interactions between the two were limited to the orientation session. The heuristic evaluations were conducted independently from the test conductor. Consequently, the test conductor being "out of sight, out of mind" may have adversely affected the effort invested in the evaluations.

Implications. Performing a heuristic evaluation of a shared visual workspace groupware with the proposed heuristics and one inspector will uncover, on average, a relatively low percentage of the teamwork problems (~20%). As a result, we cannot rely on a single inspector to uncover many problems. This does not affect my basic conclusion: that the individual inspector performance is in line (albeit somewhat lower) when compared to traditional heuristic evaluation. An initial reaction may be that it is inefficient to conduct a heuristic evaluation with one evaluator in a usability engineering project. However, these numbers are reasonable since uncovering a proportion of problems is better than finding none at all. To that end, the groupware heuristics did uncover usability bugs while exhausting minimal resources: only one individual with limited knowledge in CSCW, little training (1 hour), and minimal time and effort. Make no mistake; evaluating an interface under these circumstances will produce less than optimum results. However, given the typically tight constraints of software development projects this approach

provides a sizable return on investment for the limited cost. To improve these results, one can supplement this inspection method with other evaluation techniques (i.e., usability testing) and/or involve multiple inspectors to increase the number of problems found.

More generally, the results from the study imply that the effectiveness of the heuristic evaluation methodology is dependent on the dedication of the individual performing the inspection. Investing a minimal amount of time and effort when conducting a heuristic evaluation will produce poor results even by knowledgeable inspectors. I suspect that this is the case for all types of heuristic evaluation methodologies.

5.2.1.2 Proportion of inspectors who found each problem

In addition to analyzing the ability of the inspectors to find all known usability problems, I also compare their ability to find individual problems. In turn, I can determine if problems are equally likely to be found by many inspectors, or if some problems are easier (or harder) to find. This may also help to further explain the difference in performance between the two types of evaluators.

Results. Figure 5.3 (and Table 5.3) and Figure 5.4 (and Table 5.4) summarize the average number of inspectors who found each teamwork usability problem for GroupDraw and Groove (note, the numbered entries in the tables correspond to the numbered teamwork problems in Appendix B). For each problem, this average is calculated for each inspector type and a combination of the two. The bold entries in each table reflect a difference of 20% or more in the probability of a usability problem being found when comparing the novices to the regular specialists. Based on the combined results for all inspectors, the proportion of inspectors uncovering particular problems ranges from 4% to 68% for GroupDraw and 4% to 93% for Groove.

Discussion. For both interfaces, there is tremendous variation between the ability of the inspectors to uncover individual teamwork usability problems. As a result, one can speculate that some usability problems are easier to find while others are more difficult to uncover. Similarly, Nielsen (1992) reports that less than 15% of inspectors doing a traditional heuristic evaluation found the hardest problem in a telephone operated

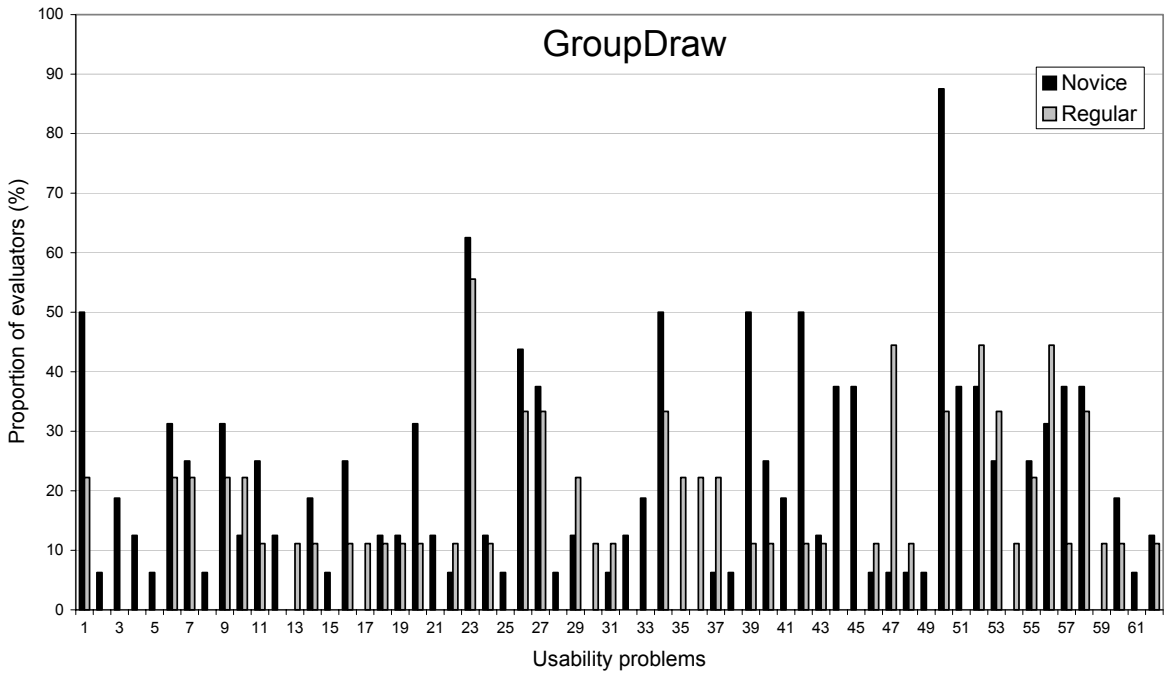

Figure 5.3 Proportion of evaluators who found each GroupDraw problem

Problem	Novice	Regular	All	Problem	Novice	Regular	All
1	**50.0**	**22.2**	40.0	32	12.5	0.0	8.0
2	6.3	0.0	4.0	33	18.8	0.0	12.0
3	18.8	0.0	12.0	34	50.0	33.3	44.0
4	12.5	0.0	8.0	**35**	**0.0**	**22.2**	8.0
5	6.3	0.0	4.0	**36**	**0.0**	**22.2**	8.0
6	31.3	22.2	28.0	37	6.3	22.2	12.0
7	25.0	22.2	24.0	38	6.3	0.0	4.0
8	6.3	0.0	4.0	**39**	**50.0**	**11.1**	36.0
9	31.3	22.2	28.0	40	25.0	11.1	20.0
10	12.5	22.2	16.0	41	18.8	0.0	12.0
11	25.0	11.1	20.0	42	50.0	11.1	36.0
12	12.5	0.0	8.0	43	12.5	11.1	12.0
13	0.0	11.1	4.0	**44**	**37.5**	**0.0**	24.0
14	18.8	11.1	16.0	**45**	**37.5**	**0.0**	24.0
15	6.3	0.0	4.0	46	6.3	11.1	8.0
16	25.0	11.1	20.0	**47**	**6.3**	**44.4**	20.0
17	0.0	11.1	4.0	48	6.3	11.1	8.0
18	12.5	11.1	12.0	49	6.3	0.0	4.0
19	12.5	11.1	12.0	**50**	**87.5**	**33.3**	68.0
20	**31.3**	**11.1**	24.0	**51**	**37.5**	**0.0**	24.0
21	12.5	0.0	8.0	52	37.5	44.4	40.0
22	6.3	11.1	8.0	53	25.0	33.3	28.0
23	62.5	55.6	60.0	54	0.0	11.1	4.0
24	12.5	11.1	12.0	55	25.0	22.2	24.0
25	6.3	0.0	4.0	56	31.3	44.4	36.0
26	43.8	33.3	40.0	**57**	**37.5**	**11.1**	28.0
27	37.5	33.3	36.0	58	37.5	33.3	36.0
28	6.3	0.0	4.0	59	0.0	11.1	4.0
29	12.5	22.2	16.0	60	18.8	11.1	16.0
30	0.0	11.1	4.0	61	6.3	0.0	4.0
31	6.3	11.1	8.0	62	12.5	11.1	12.0

Table 5.3 Proportion of evaluators who found each GroupDraw problem

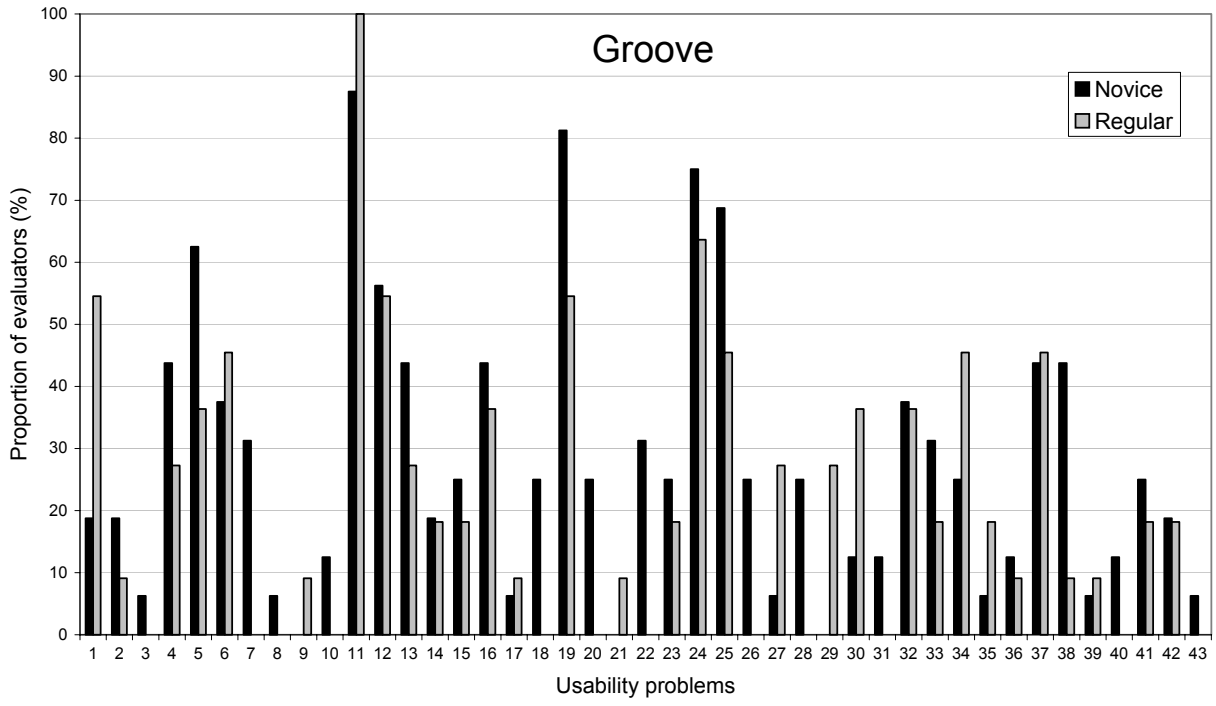

Figure 5.4 Proportion of evaluators who found each GroupDraw problem

Problem	Novice	Regular	All	Problem	Novice	Regular	All
1	**18.8**	**54.5**	33.3	23	25.0	18.2	22.2
2	18.8	9.1	14.8	24	75.0	63.6	70.4
3	6.3	0.0	3.7	**25**	**68.8**	**45.5**	59.3
4	43.8	27.3	37.0	**26**	**25.0**	**0.0**	14.8
5	**62.5**	**36.4**	51.9	**27**	**6.3**	**27.3**	14.8
6	37.5	45.5	40.7	**28**	**25.0**	**0.0**	14.8
7	**31.3**	**0.0**	18.5	**29**	**0.0**	**27.3**	11.1
8	6.3	0.0	3.7	**30**	**12.5**	**36.4**	22.2
9	0.0	9.1	3.7	31	12.5	0.0	7.4
10	12.5	0.0	7.4	32	37.5	36.4	37.0
11	87.5	100.0	92.6	33	31.3	18.2	25.9
12	56.3	54.5	55.6	**34**	**25.0**	**45.5**	33.3
13	43.8	27.3	37.0	35	6.3	18.2	11.1
14	18.8	18.2	18.5	36	12.5	9.1	11.1
15	25.0	18.2	22.2	37	43.8	45.5	44.4
16	43.8	36.4	40.7	**38**	**43.8**	**9.1**	29.6
17	6.3	9.1	7.4	39	6.3	9.1	7.4
18	**25.0**	**0.0**	14.8	40	12.5	0.0	7.4
19	**81.3**	**54.5**	70.4	41	25.0	18.2	22.2
20	25.0	0.0	14.8	42	18.8	18.2	18.5
21	0.0	9.1	3.7	43	6.3	0.0	3.7
22	**31.3**	**0.0**	18.5		25.0	18.2	22.2

Table 5.4 Proportion of evaluators who found each Groove problem

interface, while more than 70% of them found the ‘easiest problem (note, this is the combined result for the novices and regular specialists).

Comparing the two types of groupware inspectors, we see that they performed roughly equal with discovering the majority of the usability problems. For instance, the novices and regular specialists exhibited similar performance with respect to uncovering the ‘hard’ problems (less than 15% of inspectors found them) and ‘easy’ problems (more than 40% of inspectors found them). Despite this commonality, a finite number of problems have a significance difference in their probability of being uncovered by each type of inspector. With GroupDraw, eleven problems (in bold) had a performance difference greater than 20%. With the exception of four problems, the novice evaluators outperformed the regular specialists. Similarly, Groove exhibited fourteen usability problems with a difference of 20% or more. Again, the novices outperformed the regular specialists on the majority of these problems (10 of 14). Analyzing these problems does not indicate a trend that would suggest why the one type of evaluator would experience a higher success rate than the other.

Implications. The variation in the inspectors’ ability to uncover the groupware usability problems suggests that these problems differ considerably in how easy or hard they are to uncover. To that end, we expect some problems may even go unnoticed during a heuristic evaluation of an interface. Consequently, one should not rely solely on the heuristic evaluation methodology as a catch-all, especially with a small set of evaluators. For optimum results, alternative usability evaluation techniques (e.g., usability testing) should supplement a heuristic evaluation within a software project lifecycle.

The difference in performance between the two types of groupware inspectors does not appear to be based on discrepancies in knowledge or training. There is no visible pattern explaining the success rate variability with respect to uncovering particular usability problems. To reiterate, I suspect that the reduced level of effort by the regular specialists is the main contributor for their reduced performance in comparison to the novice evaluators. Even though further analyses will continue to break down the performance between the two types of inspectors, I will not continue to speculate on why this difference exists.

5.2.1.3 Consistency of individual inspector performance

Since the same evaluators heuristically evaluated both GroupDraw and Groove, we can compare their individual performance across the two systems. If each inspector performs consistently, then perhaps we could identify 'good' inspectors and use them instead of many 'poor' ones.

Results. Figure 5.5 is a scatterplot for the proportion of teamwork usability problems found by each evaluator for both systems as well as the best-fit linear regression line. The scatterplot includes only 25 evaluators since 2 of the regular specialists did not perform a heuristic evaluation of GroupDraw. Comparing the performance of the individual evaluators across both systems with a regression analysis provides an r^2 value of 0.63.

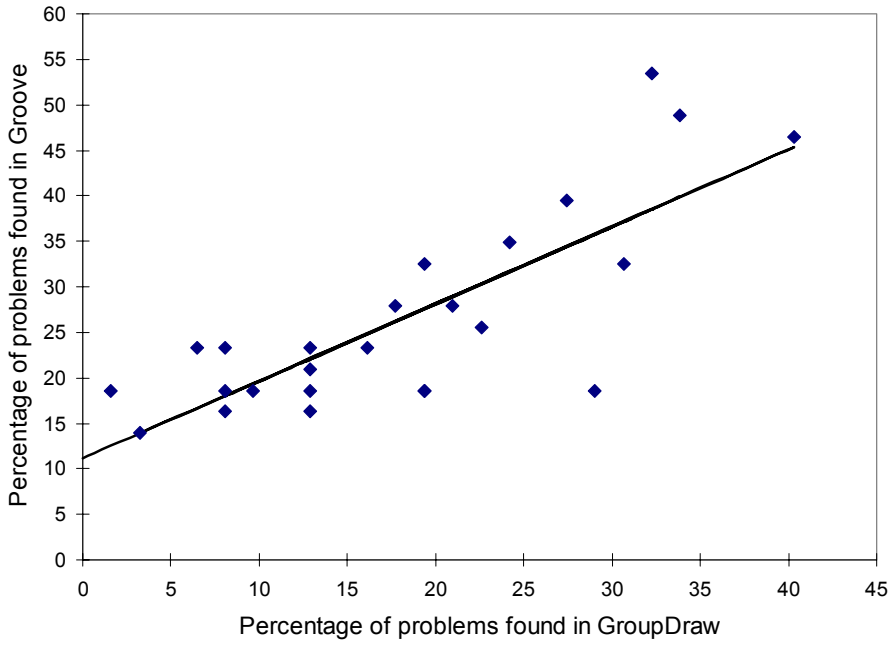

Figure 5.5 Scatterplot of the proportion of usability problems found by the same evaluators in GroupDraw and Groove

Discussion. An r^2 value equal to 0.63 suggests a modest correlation in inspectors' performance across the two systems. To illustrate, if we divide the total number of inspectors into thirds, the individuals making up the top third in terms of performance on GroupDraw are roughly the same top third performers for Groove. The same can be said

with the bottom two thirds of inspectors. However, there still exists a large amount of variability in the performance from one system to the next. Therefore, while some people are better than others at conducting heuristic evaluation of user interfaces, their performance level cannot be guaranteed from one system to the next.

The correlation results from this study differs somewhat from Nielsen and Molich's (1990) analysis of traditional heuristic evaluation, where they report only a weak correlation (r^2=0.33).

Implications. It is impractical to identify with a high degree of confidence a "good" evaluator and expect their performance to be consistently "good" for all evaluations.

5.2.1.4 Ranking of individual performance

Since some usability problems are harder to find than others and the inspectors' performance exhibits some consistency across both systems, one might assume that the evaluator performance can be ranked based on the difficulty of finding individual usability problems. In other words, do "good" evaluators uncover both the easy problems found by the "weaker" evaluators as well as the harder problems missed by the latter? Or can the "weaker" evaluators also find these harder problems?

Results. Figure 5.6 illustrates who found which usability problems for each interface. Each column corresponds to one evaluator. A column labeled 'N' signifies a novice evaluator and 'R' a regular specialist. The columns are sorted according to the number of problems found by the inspector—the 'worse' inspector on the left found few problems, while the 'best' inspector on the right found many problems. Each row represents a usability problem, where rows are sorted by how 'hard' they were to uncover—many inspectors found the 'easiest' problems on the bottom, while few inspectors found the 'hardest' problems on the top. A square is marked in black if the evaluator assigned to the column found the problem assigned to the row.

Discussion. The graphs and results are similar to Nielsen's (1992) analysis of traditional heuristic evaluation. First, the sets of usability problems found by different evaluators do not exhibit a significant amount of overlap. Most evaluators find some 'easy' usability problems, whereas some 'hard' problems are found by only a handful of evaluators. Second, even poor evaluators sometimes find hard problems as indicated by the marked

squares in the upper left corner of the figures. Finally, good evaluators may sometimes overlook easy problems as indicated by the unmarked squares in the lower right corners.

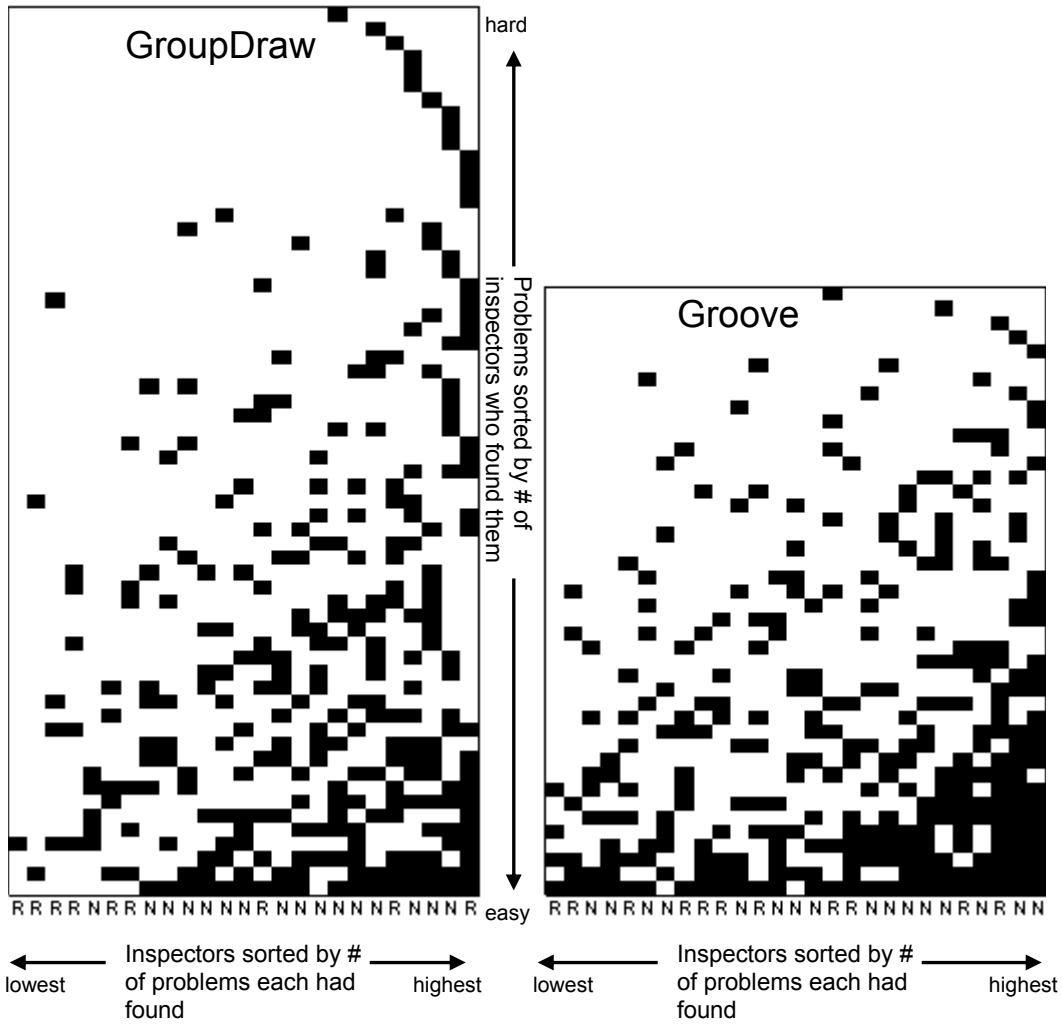

Figure 5.6 Problems found by each type of evaluator for both systems

Further to the previous discussions regarding the overall lower performance by the regular specialists, one can see how the majority of these evaluators are ranked near the bottom in terms of their performance (i.e., the majority of the columns labelled 'R' are on the left side).

Implications. Again, it appears that multiple evaluators will provide better overall performance than using a single evaluator (even if (s)he is the 'best' one). First, one

cannot just identify the 'good' evaluators and rely solely on their findings since they may overlook both 'hard' and 'easy' problems. In addition, the same individuals may not always be the best evaluator. We can also argue that even 'poor' evaluators have value, as they may find difficult problems overlooked by others. Second, the individual inspectors tend to find different problems (i.e., there is modest overlap with the individual lists of problems found). Consequently, the results suggest that the combined wisdom of multiple evaluators is better than that of the 'best' evaluator. The next section looks at the performance of multiple participants by calculating theoretical aggregated results.

5.2.2 Performance of aggregates of inspectors

How many inspectors are required to obtain the optimum results from a groupware heuristic evaluation? The naïve approach is to use as many inspectors as possible; however, this will compromise the methodology's discount status due to the increased cost and preparation time. Ideally, just a few inspectors are needed to find many problems, which would mean that the heuristic evaluation would have a good cost/benefit ratio (Nielsen and Molich 1990). To measure the cost/benefit, I formed aggregates of groupware inspectors of varying sizes to determine how many problems each one would theoretically uncover using the mechanics of collaboration heuristics.

Results. I created aggregates by collecting the teamwork usability problems found by each inspector in the aggregate to form a larger set of problems. A given usability problem was considered found if it was recorded by at least one member of the group. For each aggregate size of novice evaluators and regular specialists, I then calculated the average proportion of problems found for all possible combinations. For the 'combined' inspectors, only a representative sample of combinations for certain aggregate sizes was calculated due to the vast number of combinations. The number of combinations was limited by the fact that each individual's result could only be counted once when forming an aggregate. Figure 5.7 graphs the average proportion of problems uncovered by each size of aggregate for both GroupDraw and Groove. For comparative purposes, I superimpose Nielsen's (1992) results for 31 novice evaluators and 19 regular specialists evaluating a telephone operated interface.

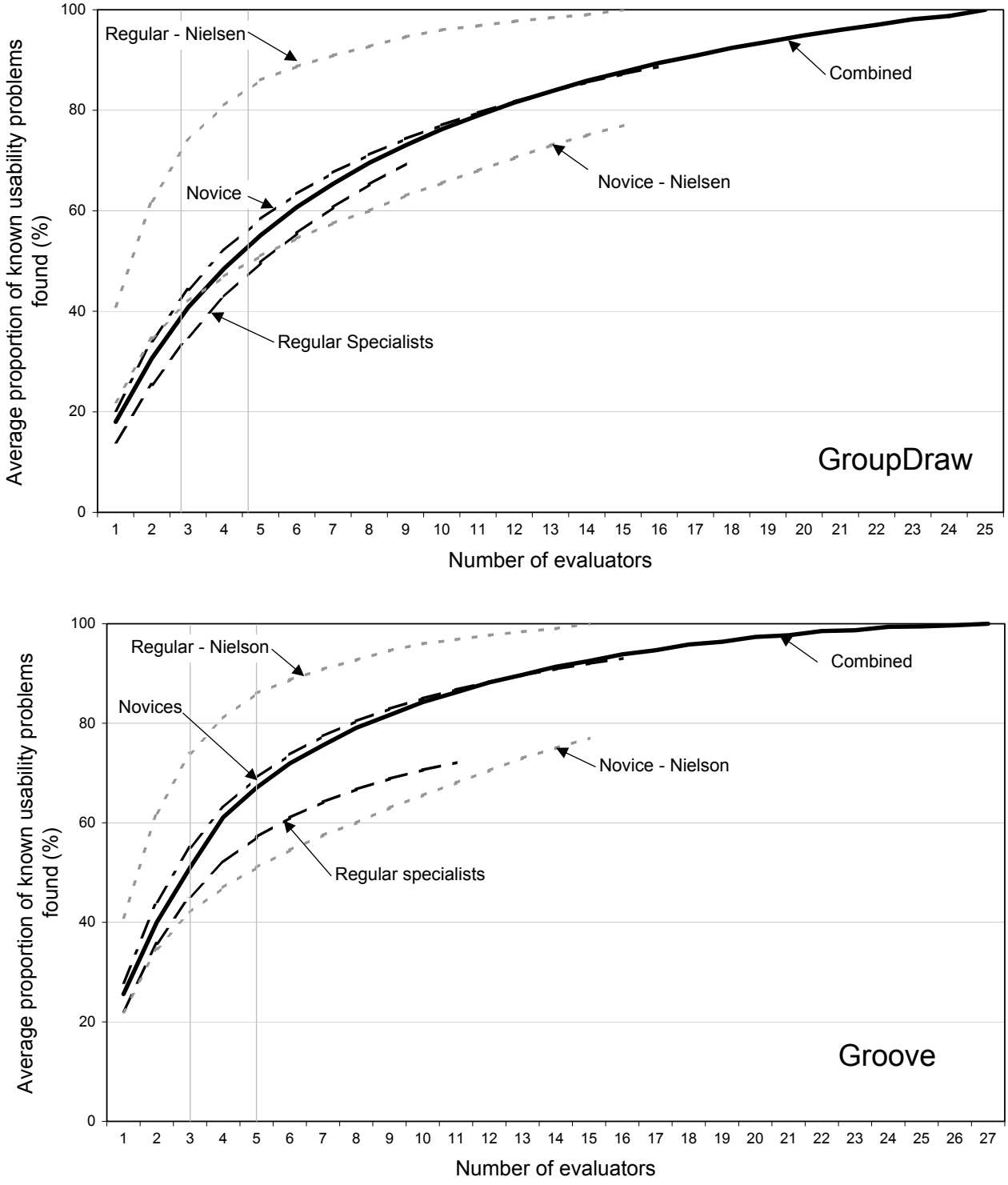

Figure 5.7 Graph of percentage of problems found by each aggregate of inspectors for GroupDraw and Groove

The shape of all lines indicates that initial performance increases rapidly as the number of inspectors increase, but then more slowly afterwards. For example, while one inspector will find an average of 20-25% of the problems in both systems, three inspectors will find 40-50% of the problems, and five will find about 50-60%. Quite a few more inspectors are required to do significantly better than that. Finding 75% of all problems, for instance, requires nine evaluators for GroupDraw and seven for Groove. As expected from the earlier analysis, the curve for the regular specialists is lower than that of the novices in both cases.

Discussion. As depicted by Figure 5.7, an aggregate of evaluators will obtain better results than a single evaluator when conducting an inspection with the groupware heuristics. However, increasing the number of inspectors also makes the method more expensive. Thus we need to identify a reasonable optimum that trades off the cost of employing multiple inspectors vs. the number of usability problems uncovered. In traditional heuristic evaluation, Nielsen recommends using between three and five usability specialists to find a reasonably high proportion (~75%) of problems in a given interface (Nielsen and Molich 1990, Nielsen 1992). Novice evaluators are not as effective—a group of five (i.e., the recommended upper range of group size) reported only 51% of problems. Using Nielsen's recommendation of three to five evaluators, the 'combined' results suggest that inspectors will find between 40-60% of the total known teamwork usability problems. This performance level provides a reasonable cost/benefit ratio. This result is also quite good considering the expense of other groupware evaluation methods. A minimal amount of time, expertise, and training were expended to achieve these results. As the heuristics evolve, people become more familiar with their content, and the incentive is present to diligently perform an evaluation, I would expect that the results to continually improve.

This performance of the groupware evaluators fits between Nielsen's novice and regular evaluators. This is to be expected since the combined results contain a mix of individuals with and without expertise in HCI and CSCW. It may be unrealistic to expect the same high level of performance by groupware evaluators as experienced by Nielsen's traditional usability specialists. In general, individuals will not typically have the same

level of expertise and experience with groupware as they do with traditional single-user interfaces. Hence, a CSCW specialist may not truly exist in the same manner as Nielsen's usability specialists.

Implications. The success of heuristic evaluation is improved if multiple evaluators are employed. The optimum number of evaluators to use will depend upon the compromise made between the increased costs of using each extra evaluator versus the cost of not identifying usability problems. My recommendation would be to perform a heuristic evaluation using three and five evaluators with additional resources spent on alternative methods of evaluation. More evaluators can be added to improve results but returns are incremental.

5.2.3 Characterizing found teamwork usability problems

This section analyses the type of problems uncovered using the groupware heuristics by categorizing them according to severity (i.e., expected impact on the users) and violated heuristic. In addition, I provide general observations regarding the applicability of the heuristics to uncovering teamwork usability problems across the two systems.

5.2.3.1 Major vs. minor usability problems

Typically, the chances of a usability problem being corrected within a tight schedule are dependent on its severity. Major problems are those that introduce serious obstacles to effective collaboration whereas minor problems can typically be worked around by the end user. Clearly, it is more beneficial if the proposed groupware heuristics are good for uncovering major problems. Therefore, I now analyze and compare the inspectors' performance for uncovering major and minor problems.

Results. Table 5.5 summarizes the inspectors' average performance for uncovering the major and minor problems in both systems.

Based on the aggregated results, the groupware heuristics uncovered virtually the same proportion of major usability problems in both systems. In GroupDraw and Groove, 26% (16 out of 62) and 30% (13 out of 43) of the total known usability problems are major, respectively. Overall, approximately 28% (29 out of 105) of all reported problems are major. This is identical to traditional heuristic evaluation where Nielsen (1992)

System	Problem category	Num. of problems	Novice (%)	Regular (%)	All (%)
GroupDraw	Major problems	16	29.7	18.1	25.5
	Minor problem	46	17.1	12.8	15.6
	Total known problems	62	20.3	14.0	18.0
Groove	Major problems	13	48.6	39.9	45.0
	Minor problem	30	18.8	14.5	17.0
	Total known problems	43	27.9	22.2	25.6
Both	Major problems	29	39.1	30.0	35.6
	Minor problem	76	17.9	13.8	16.3
	Total known problems	105	23.7	17.2	21.2

Table 5.5 Average proportions of problem types found by inspectors

reports that 28% (59 out of 211) of the found problems across six case studies were classified as major. On an individual basis, each groupware inspector found, on average, slightly more minor problems than major ones (a ratio of 3:2). In contrast, Nielsen (1992) reports a higher ratio of 2 minor problems for each major one.

When compared to the total known problems, the individual inspectors tend to uncover a higher proportion (36%) of the major usability problems versus a lesser proportion (16.3%) of the minor problems ($p<.01$). Again, these results are in line with traditional heuristic evaluation. Averaged across six studies (Nielsen 1992), inspectors found 42% of major problems versus 32% of minor problems. This difference was calculated to be statistically significant.

Discussion. These results suggest two tendencies for groupware heuristic evaluation, reflecting what is also seen in traditional heuristics evaluation (Nielsen and Molich 1990, Nielsen 1992). First, inspectors find a higher proportion of the total number of major versus minor usability problems in a given interface. This implies that major problems are easier to find than minor ones, that evaluators pay relatively more attention to finding them, and that the same problem will be reported by multiple inspectors. This is advantageous since major usability problems are by definition the most important ones to find and fix. Second, the individual as well as aggregate results indicate that inspectors

will find more minor than major problems in terms of absolute numbers for a given interface. This suggests that inspectors do not exclusively focus on major problems while neglecting the minor ones. Uncovering minor problems is still relevant since an abundance of these less severe problems can accumulate to make collaboration cumbersome. In addition, unlike major problems that are often spotted by many inspectors, fewer inspectors may notice minor problems.

Implications: The groupware heuristics are good for finding both major and minor problems in a system, with major problems being slightly easier to find than minor ones. This result is good news for the cost-benefit analysis reported in Section 5.2.2, for it suggests that while 3-5 evaluators will uncover 40-60% of the usability problems, there is a good chance that many of these uncovered problems will be major, and that most missed problems will be minor. However, minor problems are still relevant. In fact, Nielsen (1994a) discovered that minor problems appear easier to find by heuristic evaluation than by other methods. If the same holds true for groupware heuristic evaluation, then this methodology would complement other groupware evaluation techniques (e.g., field studies). Further research is required to validate this hypothesis. Finally, since more minor problems are found by the groupware heuristic, severity ratings play an important role since inspectors can prioritize which problems should be fixed.

5.2.3.2 Exploratory coverage of each heuristic

Since heuristic evaluation involves judging interfaces against established usability principles, this section explores whether problems violating certain heuristics are easier to find than others.

Results. Table 5.6 (and Figure 5.8) classifies the usability problems and the average performance of individual evaluators according to the eight mechanics of collaboration heuristics. All the inspectors perform roughly equal across all the heuristics with the average performance ranging from ~18% to ~29%. The exception is heuristic 2 where the evaluators had an average performance of ~53%. In this case, the inspectors performed better due to the single Groove usability problem violating this heuristic. This problem sums up the fact that the system did not adhere to this heuristic at all (i.e. the system did not provide any means for supporting intentional gestures). Accordingly, a high

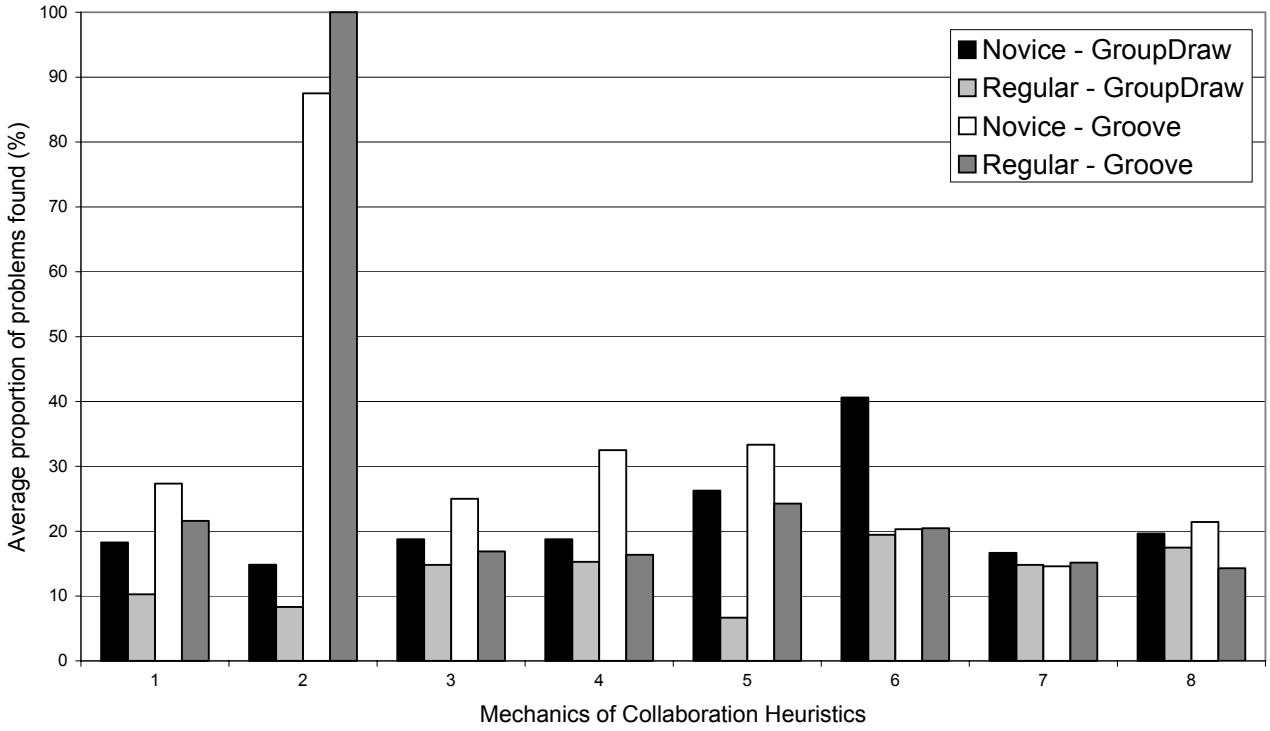

Figure 5.8 Average proportion of problems found according to the heuristics

Heuristic	GroupDraw				Groove				Both
	Num. of problems	Novice (%)	Regular (%)	All (%)	Num. of problems	Novice (%)	Regular (%)	All (%)	All (%)
1	13	18.3	10.3	15.4	10	23.8	18.2	21.5	18.4
2	8	14.8	8.3	12.5	1	87.5	100.0	92.6	52.5
3	12	18.8	15.7	17.7	7	26.8	18.2	23.3	20.5
4	8	19.5	15.3	18.0	5	32.5	16.4	25.9	22.0
5	7	22.3	12.7	18.9	6	33.3	27.3	30.9	24.9
6	4	42.2	19.4	34.0	4	23.4	22.7	23.1	28.6
7	3	16.7	22.2	18.7	3	14.6	24.2	18.5	18.6
8	7	20.5	17.5	19.4	7	22.3	14.3	19.0	19.2

Table 5.6 Problems classified according to the mechanics of collaboration heuristics

percentage of the inspectors reported this glaring non-compliance. Similarly, Nielsen (1992) found the evaluators' average performance to be roughly equal across his nine original heuristics (excluding "help and documentation"). Although, Nielsen reports that

it is harder to find usability problems violating three of the heuristics when compared to the other six.

Discussion. The evaluators' consistent performance across the eight groupware heuristics suggests that usability problems identified during a heuristic evaluation have about the same probability of being found for most of the heuristics. I attribute the difference in the number of problems found by each heuristic to be simply a result of the system containing more problems related to the heuristic. I do not view it as a sign that the inspectors are experiencing difficulties applying certain individual heuristics. As a footnote, the classification of the problems according to heuristics can be somewhat arbitrary. Specifically, a finite number of problems may be classified under multiple heuristics due to the heuristics' overlap in coverage. For instance, I came across instances during the results synthesis where different inspectors labeled identical problems according to different heuristics. From a practical standpoint, this overlap is beneficial since it improves the identification of usability problems. In addition, classifying the problems according to the 'correct' heuristic is not crucial to the methodology's success.

To further assess the inspectors' ability to apply each groupware heuristic, I analyzed the feedback forms filled out by the inspectors assessing the ease with which they were able to comprehend the heuristics. For the most part, the inspectors did not express any significant difficulties. However, they did raise numerous questions regarding the content of heuristic #7, Support people with the coordination of their actions. Specific comments included:

> *"Too much related to previous heuristics. What should we measure here?"*
>
> *"I am not exactly clear about the distinction between this heuristic and heuristic #5. I see some of the differences, but it's probably the fact that this heuristic 'bundles up many previous heuristics' that I am uncertain whether or not I could use it effectively to analyze an interface."*

This heuristic is at a higher-level than the others since it bundles up many of the fundamental mechanics described in the other heuristics. Therefore, many inspectors had difficulties understanding how to apply this heuristic and how it differed from the others (especially heuristic 5, Provide protection).

Implications. Even though the inspectors exhibited roughly the same level of performance across the eight heuristics, I suspect that heuristic 7 requires further revisions to address the user feedback and improve its practicality. At this point, additional measures may be necessary to find problems relating to this heuristic.

5.2.4 Usability problems across the two systems

GroupDraw and Groove are quite different systems. GroupDraw was designed by a Grouplab researcher who was familiar with much of the groupware literature that inspired the mechanics of collaboration. Consequently, even though it is a toy editor, GroupDraw adheres to many of the fundamental principles behind the mechanics of collaboration heuristics. Groove, on the other hand, is a commercial product. While it looks far more professional than GroupDraw, it became quite clear that it did not adhere at all to some of the heuristics. Because of this difference, we wanted to understand how inspectors were able to apply the groupware heuristics to both systems in terms of the nature of the problems identified. To explore this issue, I examined the problem reports generated during the inspection of the two systems.

GroupDraw typically supported the partial requirements of each heuristic. Consequently, inspecting the system's compliance to the each heuristic was not necessarily 'black and white'; the inspector was required to probe a little deeper to determine to what extent the heuristic was supported. For instance, GroupDraw's telepointers support intentional gestures (heuristic 2) and an individual's embodiment (heuristic 3). However, heuristic 2 also encourages deictic references, and the lack of a verbal channel in GroupDraw means that this is hard to do. Similarly, studying heuristic 3 would reveal that all the telepointers are identical making it impossible to differentiate one from the other. This description of the GroupDraw's partial non-compliance with the mechanics of collaboration was typical of most its problem reports.

In contrast, Groove failed to comply with certain heuristics at the most basic level. For example, the Outliner tool had no means whatsoever to support intentional gestures (heuristic 2). Consequently, problem reports were often simple blanket statements about a non-compliance, such as "The system does not support intentional gestures". (This also explains the lesser number of reported problems in Groove *vs.*

GroupDraw in Table 5.2). Although this made the results synthesis cumbersome, the heuristics were essential to pointing out the fundamental flaws with the system's ability to support collaboration within a shared workspace. However, once this is pointed out there is not much else an inspector has to say about the interface. In addition, these types of problem reports do not lend themselves to providing specific design recommendations for improving the system.

5.3 Locales Framework – Analysis and interpretation

As a complement to the mechanics of collaboration heuristics, the regular specialists also inspected both GroupDraw and Groove with the Locales Framework heuristics. This section explores the practicality of this second set of groupware heuristics. Since the Locales Framework heuristics are not the main thrust of this research study, I have limited my assessment to a qualitative one based on:

- my own personal experience re-writing the original heuristics;
- the inspectors' feedback from questionnaires and post-test discussions; and
- the usability problems uncovered during the heuristic evaluation of GroupDraw and Groove.

I then present the implications that these findings have on the current version of the Locales Framework heuristics.

5.3.1 My personal experience

Greenberg et al. (2000) limit the original five Locales Framework heuristics to one or two sentences. For the purposes of their study, their brevity is adequate since the inspectors applying the heuristics are all intimately familiar with the Locales Framework. However, the average inspectors will typically not have prior experience or expertise with this framework. Consequently, the heuristics' brevity makes them impractical for conducting an effective heuristic evaluation of groupware in these situations. To that end, I set out to elaborate each heuristic with the purposes of making them easier to understand. Even though I maintain the same heuristics as Greenberg et al. (2000), this task was still a significant undertaking. First, I had to learn the nuances of the Locales Framework

(Fitzpatrick 1998). In comparison to the mechanics of collaboration, the underlying concepts supporting this framework are more abstract and subtle, and consequently more difficult to grasp. Next, I expanded each heuristic by adding theory and concrete examples. The goal was to improve their comprehension and hence their practicality (see Appendix A.6 for the resulting heuristics). To clarify concepts surrounding each heuristic, I had numerous discussions with an author of the original heuristics. Given my experience with trying to comprehend and subsequently revise the Locales Framework heuristics, I suspect that these heuristics will typically require more time and energy for an inspector to understand and apply them.

The training sessions with the regular specialists reflected my personal difficulties with the Locales Framework. A significant amount of time was spent explaining and clarifying many of the concepts described in the written materials.

5.3.2 Evaluator feedback

I obtained evaluator feedback on the utility of the heuristics from the pre- and post-training questionnaires (see Appendix A.4). A review of the completed questionnaires revealed some interesting issues surrounding each heuristic. As I suspected, written comments reflect a fundamental difficulty with understanding some of the Locales Framework principles. Interestingly, many of the inspectors still rated that they understood the concepts and would have no problem applying the heuristics during an evaluation. General comments on the Locales Framework heuristics include:

> *"I am very familiar with these concepts, so I know these heuristics. However, I suspect that some of the ideas are 'fuzzy' for someone who doesn't know the background literature."*
>
> *"I no longer feel that I can effectively apply the heuristics, even after trying to apply them. Moreover, I am not entirely convinced that this is just another case of me being stupid."*

The first comment highlights the overall struggles inherent to understanding the theory deriving each heuristic. In turn, the latter stresses the difficulties applying the heuristics to uncover usability problems in Groove and GroupDraw.

In addition to these general comments, the evaluators recorded comments specific to each heuristic. These comments help to further break down where people had

difficulties understanding concepts pertaining to each individual heuristic. The first heuristic, Provide Locales, had numerous comments including:

> *"I roughly understand and could probably use it to do a crude analysis of an interface, but I am certain there may be significant things I will miss due to incomplete understanding."*
>
> *"Too general. What should we evaluate here?"*
>
> *"I am sure I can recognize if there is a locale, but I am not sure I can suggest how to add or improve a locale."*
>
> *"It is difficult to separate the tangible space (virtual or physical) with the conception of site."*
>
> *"Virtual 'site' but real world 'means'. Confusing. Can these be broken up?"*
>
> *"I don't quite understand how this heuristic, in application, is different from most of the groupware heuristics in the previous section."*

These comments suggest that the inspectors encountered difficulties understanding how these concepts can be applied to uncovering design deficiencies in groupware. Of all the Locales Framework heuristics, understanding this first heuristic is probably the most critical. Many of the fundamental concepts of the framework (e.g., locales, means, social world) are introduced here and subsequently carried over to the other heuristics.

Comments of this nature were not limited to the first heuristic. The four other heuristics also encountered similar problems regarding their comprehension and practicality. For instance, one inspector referred to heuristic 3 as "tough" and another inspector "was lost" on heuristic 4.

Finally, responses from the post-training questionnaires reinforced the benefits of the training session for the Locales Framework heuristics. The session helped to clarify confusing issues surrounding the written version of the heuristics. For example, two post-training comments are:

> *"Now I understand locales heuristics are in a more global environment – not just the system."*
>
> *"Training session really helped to address problems with written documentation."*

Overall, the evaluator feedback reflects the need to further re-write and improve the current version of the locales heuristics.

5.3.3 Usability problems uncovered

In addition to the evaluators' feedback, analyzing the usability problems uncovered during the heuristic evaluation helps to further assess the practicality of the Locales Framework heuristics. (I do not perform an in-depth analysis of these problems as done with the mechanics of collaboration heuristics since this is not the focus of my research.)

In terms of absolute number of problems found, the Locales Framework heuristics identified numerous problems for each interface (14 for GroupDraw and 28 for Groove). However a portion of these problems overlap with those reported by the mechanics of collaboration heuristics (2 for GroupDraw and 8 for Groove). Although, this redundancy is beneficial, it demonstrates two phenomenons:

1. Individuals have a better grasp on applying the mechanics of collaboration heuristics since they had difficulty looking beyond the mechanics to the bigger scope offered by the Locales Framework heuristics (they were instructed to identify only those problems that were not captured by the mechanics).
2. In this study, the Locales Framework heuristics provided minimal added value above and beyond the mechanics of collaboration heuristics.

Despite the number of problems identified by the inspectors with the Locales Framework heuristics, there was minimal overlap between the various inspectors. In many cases, only one or two inspectors reported a given usability problem. By extrapolation, it would take a large aggregate of inspectors to uncover a substantial proportion of the total known usability problems in either system with these heuristics. Naturally, this raises the costs for conducting this type of evaluation.

The inspectors typically described the usability problems identified by the Locales Framework heuristics in general terms. For instance, one inspector reported that GroupDraw had "no planning mechanism." Similarly, another stated that Groove "did not provide historic information". Conversely, the inspectors were able to identify specific interaction problems with the mechanics of collaboration heuristics. For example, both systems "did not display the intermediate steps of all actions". While the Locales Framework heuristics identified relevant global problems with each system, I suspect that

it would be more difficult to generate fixes given the non-descript explanation of the usability problems.

Finally, unlike the set of usability problems derived by the inspectors using the mechanics of collaboration, I have personally identified numerous problems that were missed by everybody.

5.3.4 Implications

I concur with Greenberg et al. (2000) in that the Locales Framework provides a feasible backbone for developing heuristics for the purposes of identifying usability problems in complex groupware systems. After all, the heuristics are grounded in a proven theory for understanding group collaboration (Fitzpatrick 1998). In addition, these heuristics do uncover complementary usability problems to those reported by the mechanics of collaboration. Whereas the mechanics of collaboration focus on the low-level actions performed by individuals in a shared workspace, the Locales Framework steps back and focuses on the social issues surrounding teamwork. As identified in Chapter 1, these social issues can have a profound effect on the success of any groupware system.

Having said this, I still find that the Locales Framework heuristics in their current revised format to be inadequate in their utility. The nuances surrounding each heuristics make them difficult to understand and consequently difficult to apply. This contradicts one of Nielsen's mandates for his traditional heuristic evaluation: to make it easy to understand. Greenberg et al.'s (2000) success in using these heuristics to identify problems in TeamWave is due to the evaluators' expertise with the Locales Framework and groupware in general. In reality, finding evaluators with this level of expertise to conduct a heuristic evaluation with the Locales Framework heuristics is highly unlikely (and also expensive). Subsequently, if the Locales Framework is going to be used as a basis for a set of heuristics, they will require significant revisions to improve their practicality.

5.4 Summary interpretation of results

In this section, I sum up the results from the separate analyses in order to address the main objective of the research study: Do the proposed groupware heuristics allow us to evaluate groupware systems in a reasonable manner? In response to this question, I provide one answer for the mechanics of collaboration heuristics and another for the Locales Framework heuristics.

Overall, the mechanics of collaboration heuristics provide an effective and cost-efficient means for identifying teamwork usability problems related to real-time collaboration within a shared workspace. First, the lengthy consolidated list of problems for each system, as derived by the results synthesis (Section 5.1.2), illustrates the effectiveness of the heuristics. At this point I raise two concerns that bring into question this effectiveness—the validity and completeness of the reported problems. However, I defer discussion of these issues to the next section. To further assess the heuristics' cost-efficiency (i.e., practicality), section 5.2.1 demonstrates that on average, individual evaluators (based on the aggregated results for both novices and regular specialists) will uncover roughly one-fifth of the total known teamwork usability problems for two separate systems. In addition, I found that the evaluators exhibit a minimal amount of overlap in their performance. Therefore, the heuristic evaluation will produce better results if several people are used. To that end, section 5.2.2 studies the performance of aggregated inspectors. Results indicate that aggregates of 3 to 5 evaluators will uncover between 40 and 60 percent of the total know teamwork usability problems. The precise numbers of evaluators to use will ultimately depend on a trade-off between the costs of using additional inspectors versus the cost of leaving problems unfound. This level of individual and aggregate performance with the mechanics of collaboration heuristics is quite promising since minimal resources were expended in performing the evaluation (especially in comparison to existing groupware evaluation methods):

- evaluators had limited training with the heuristics – one hour session
- the majority of evaluators had limited or no expertise in groupware
- all evaluators, with the exception of one, had no previous exposure with the revised heuristics

- each evaluator dictated the amount of time and effort spent on the evaluation

I am convinced that the inspectors will do even better if any of these factors are improved (e.g., better training, more practice evaluating groupware). I also believe that the current version of the mechanics of collaboration heuristics could be improved to make them easier to learn and apply. They could also be fine-tuned or added to provide even better coverage of potential teamwork usability problems. Like Nielsen's heuristics, the proposed groupware heuristics with evolve over time.

Finally, I analyzed the types of usability problems uncovered by the mechanics of collaboration heuristics in section 5.2.3. The heuristics uncovered a larger number of minor teamwork problems in both systems. In addition, the inspectors reported a larger proportion of the total know major usability problems. Therefore, the revised heuristics do provide useful results by identifying quality usability problems. I also assessed the ability of the evaluators to identify problems with each heuristic. The results suggest that usability problems have about the same probability of being found in a heuristic evaluation for most of the heuristics. All the results stemming from this research study are analogous with Nielsen's findings for his traditional heuristic evaluation. This helps to further substantiate the practicality of the mechanics of collaboration heuristics.

With respect to the Locales Framework heuristics, they are not as successful as the mechanics of collaboration. Even though they do present a complementary set of heuristics to the mechanics and they are based on a proven theory regarding successful teamwork, their practicality is limited. This is due to the subtle nuances surrounding each heuristic which makes them more difficult to understand and hence more difficult to effectively use for identifying usability bugs in groupware by non-experts.

5.5 Discussion

There are a number of issues surrounding the methodology and results that warrant further discussion. First, I will discuss concerns surrounding Nielsen's methodology and how this affects my results. Next, I address some of the lessons learned as I reflect back

on the whole experiment. Despite these issues with the methodology, I am still confident with the general findings stemming from this study.

5.5.1 Nielsen's methodology

Upon employing a methodology similar in many ways to Nielsen's approach for validating his heuristics, I find it prudent to critique his methodology and discuss any consequential fallout.

5.5.1.1 Recognized shortcomings

As part of his case studies, Nielsen addresses potential shortcomings with his methodology. Specifically, he discusses issues regarding the validity of the reported usability problems and the completeness of the problem sets for each evaluated interface. Since I replicated Nielsen's methodology, I will now outline these two issues and explain how I cope with their contributing uncertainties.

Molich and Nielsen (1990) raise the concern whether the usability problems reported by the inspectors are in fact true usability problems that real end users would encounter with the system in the real world. Empirical evaluations were not conducted to corroborate the validity of the reported problems. This concern naturally extends to the reported teamwork usability problems for GroupDraw and Groove. To substantiate their validity as 'true' usability problems, I make two arguments similar to Nielsen and Molich (1990). First, the two proposed sets of groupware heuristics are based on well-documented frameworks regarding the real-time collaboration of small groups within a shared workspace (Gutwin 1997, Fitzpatrick 1999). Consequently, the reported problems violate established knowledge and principles outlining these types of interactions. Second, both systems are not designed for specialized users with a particular expertise or training. Consequently, any inspector evaluating the system represents a potential end user. In this research study, 25 and 27 'end users' worked their way through GroupDraw and Groove respectively. Ultimately, I believe (similar to Nielsen) that the reported problems are in fact real teamwork usability problems.

The degree with which the reported usability problems constitute the complete set of problems for each interface is also a natural concern (Nielsen and Molich 1990).

Again, the same concern applies to the problem sets derived from the GroupDraw and Groove evaluations. It is impossible to say with absolute certainty that every single teamwork usability problem was reported for both systems. A new inspector may uncover something completely new. To address this issue, I approach my analyses in a similar fashion to Nielsen (1992) by reporting the total number of "known" teamwork usability problems for each system and calculating the inspectors' relative performance. Consequently, the results suggest performance trends—a single inspector will find a low proportion of the total number of problems in a given interface and multiple individuals will find more. Given the fact that a large number of inspectors evaluated both groupware systems, I am confident that the bulk of *major* teamwork usability problems were reported and that unreported problems (which I suspect are rather few) will tend to be *minor*. In addition, I neither suggest nor expect the heuristic evaluation methodology to catch all problems in a system; it should be used in conjunction with other evaluation techniques to find the largest set of problems. Finally, unlike single user interfaces and traditional heuristic evaluation, there are numerous other factors (i.e., social, organizational, technical) that reside outside the area of coverage proposed by these groupware heuristics.

5.5.1.2 Critique

In addition to Nielsen's acknowledged shortcomings, I have two personal concerns with his methodology—the categorization of his different types of inspectors and the validity of his statistical analysis.

One issue that came to my attention while replicating Nielsen's work is the seemingly inconsistent categorization of his inspectors. In an early study, Nielsen and Molich (1990) state that "to test the practical applicability of heuristic evaluation, we conducted four experiments where people who were not usability experts analyzed a user interface heuristically". To test this hypothesis, Nielsen uses computer science students as subjects to evaluate three interfaces. All students were taking a class in user interface (UI) design and had received a training session on the heuristics prior to the evaluation. In a later study, Nielsen (1992) employs a categorization scheme to compare the inspectors' performance based on their level of UI expertise. The first category was the

novices—computer science students who had completed their first programming science course but had no formal knowledge of UI design principles. Next were regular usability specialists—people with experience in UI design and evaluation based on graduate degrees and/or several years of job experience in the usability area. Similarly, Nielsen (1994a) uses regular specialists—people "employed with usability as the only or major part of their job description and had an average of seven years experience in the HCI field"—for evaluating a system. Based on the categories and their criteria, I equate Nielsen's inspectors from his earlier 1990 study to be novices. Yet, when Nielsen (1992) performs an analysis based on the amalgamated results across six case studies, he groups the 1990 computer science students (i.e., novices) with the 1992 and 1994a regular specialists. He justifies this grouping by saying that the 1992 regular specialists are the closest to the evaluators used in the other studies. This grouping appears inaccurate due to the inconsistencies with the levels of UI expertise between the inspectors from the various case studies. Nielsen does not employ strict methods for defining the inspector categories. My methodology suffers from the same shortcoming. To circumvent I could employ measures to improve inspector categorization as will be discussed in the next section.

Nielsen performs numerous statistical comparisons to study the causal effect that factors such as UI expertise and problem severity have on the inspectors' ability to uncover usability problems. A good example is Nielsen's (1992) claim that the difference in performance between inspectors with varying levels of UI and domain expertise (i.e., novices vs. regular specialists vs. double specialists) is statistically significant. While I agree with general trend between the different types of evaluators (e.g., regular specialists perform better than novices), I have trouble believing we can statistically analyse the data and provide—with confidence—these objective conclusions. While performing his case studies, Nielsen does not incorporate rigorous controls (at least he doesn't talk about it) that I personally deem as a prerequisite to performing an analysis of this nature. For instance, the difference in evaluator performance could be a result of a myriad of factors such as inspectors spending more time on the evaluation or performing the evaluation with more diligence. Given this level of uncertainty, I did not perform a similar set of

statistical analysis. If I were to analyse the data at this level I would introduce tighter controls (refer to Section 5.5.2) to ensure that potential confounding variables do not affect the performance.

5.5.2 Critique of my methodology

Upon completing the research study, I reflect back on the methodology and discuss some of my lessons learned. Although my goal was not to run a tightly controlled experiment, I provide recommendations on how one could improve the experiment in this respect.

In conducting the groupware heuristic evaluation with both types of evaluators, my aim was to assess the effect that CSCW expertise had on the evaluators' ability to uncover usability problems with the heuristics. I expected the regular specialists to out-perform the novices based on their added experience in CSCW. However, the exact opposite occurred as the novice evaluators produced better results than the regular specialists. Earlier in Section 5.2.1.1, I speculate that two reasons for this outcome were the unequal time commitment and incentive between the two groups. To address these two variables, I could have done the following:

1. Unlike the novice evaluators, the regular specialists evaluated GroupDraw and Groove with the two sets of heuristics. Looking back, I should have limited their time commitment to performing the heuristic evaluation with only the mechanics of collaboration heuristics. The Locales Framework heuristics were a secondary objective for my research study. The amount of time invested by the inspectors performing this task did not provide an equivalent return on investment. If anything, the increased workload hindered their evaluation of the systems with the mechanics of collaboration heuristics.
2. The novice evaluators performed the experiment as part of their course, whereas the regular specialists participated on a purely voluntary basis. One of the caveats with performing a "good" heuristic evaluation is the evaluator's level of commitment. Possible means for ensuring the same motivation for all participants would be to 'hire' inspectors as contractors and provide monetary remuneration.

If the primary objective of my study was to assess the relationship between CSCW expertise and inspector performance, I could have imposed other tighter control

on the experiment. In addition to the two aforementioned controls, a time restriction could have been introduced to ensure that all inspectors, regardless of their classification, performed the heuristic evaluations under identical conditions. Although, I suspect that this would have restricted the other conclusions (e.g., regarding individual and aggregate performances) that I could make. Since the scope of my research study was larger than simply evaluating this relationship I chose to forgo a time limit on the evaluations. I could also elaborate the criteria used to profile the inspectors. The background questionnaire, for instance, could have used coding questions to score the different inspectors as well as elicit people's experiences with groupware. This would help to further categorize the inspectors as opposed to relying solely on their educational background as the defining factor. Controlling these variables in this manner may have produced more comparable results between the two types of evaluators.

The inspectors performed the heuristic evaluations through self-guided exploration. As mentioned earlier, I chose this route since both systems were designed for use by the general population, hence the evaluators did not require scenarios to assist them. In addition, this represents the typical process for conducting a heuristic evaluation in the "real world". However, some evaluators indicated that it would have been easier to perform the evaluation if they had a goal or purpose for using the system as opposed to an ad hoc approach. This was especially true for Groove. Groove is a large-scale system and individuals were easily sidetracked. To illustrate, I received these comments:

> *"Although I realize that our assignment was not a usability study, having a specific task to do might have helped. I could not see any apparent point to the Outliner tool. Because this tool seemed so useless, we quickly were tired of it, and started trying to get something else to work …"*
>
> *"…Groove was very large and was very poor. The group of people that I tested it with got lost very easily in it. If the tasks given were more concrete it might help narrow the evaluation down to the specific area."*

To add control to the experiment, supplying specific scenarios would ensure that the inspectors concentrated on the exact same functionality during their inspection. This would also help explore the relationship between inspector expertise and performance.

One of the strengths of Nielsen's traditional heuristic evaluation is the fact that different inspectors uncover different usability problems. As a result, pooling together the

independent results from multiple inspectors creates substantially better results. Inspectors working together during their evaluation might bias each other towards a certain way of approaching the analysis. This may in turn limit the variety of usability problems uncovered. Even though they were instructed to record usability problems independently from one another (refer to section 4.3.3), the inspectors had to work in pairs in order to thoroughly exercise the systems' functionality. In many instances, I could identify which pair of individuals performed the inspection together since their problem sets were similar. This limited the variety of problems uncovered. Unfortunately, given the nature of groupware, it is difficult to overcome this shortcoming when performing heuristic evaluation of collaborative applications such as GroupDraw and Groove.

5.6 Conclusion

This chapter provides an in-depth analysis of the heuristics based on the list of usability problems derived from the results synthesis. Overall, the mechanics of collaboration heuristics are suitable for typical practitioners to uncover usability problems specific to real-time, shared workspace groupware. Unfortunately, the Locales Framework heuristics require more improvement and subsequently more research to provide the same level of practicality. In the following chapter, I sum up the entire research study as I state how I satisfied by research objectives, as well as provide some ideas for future research in relation to this study.

Chapter 6: Summary

This research study has explored the creation and validation of two sets of heuristics for the purposes of inspecting groupware systems for teamwork usability problems via a heuristic evaluation. The motivation behind this research is that current real-time distributed groupware systems are awkward and cumbersome to use, a situation partly caused by the lack of practical groupware evaluation methodologies. As a researcher and practitioner, I created the groupware heuristics with the goal of capitalizing on Nielsen's popular heuristic evaluation methodology as a means to evaluate the ability of groupware systems to support collaborative activities.

This chapter concludes the research study and has four parts. First, I revisit the research objectives set out in Chapter 1, where I summarize how those objectives were met, and assess my progress on each objective (Section 6.1). Second, I summarize the major and minor original contributions that this research has made on CSCW research (Section 6.2). Third, I describe directions for future work based on the research done here (Section 6.3). Finally, I touch upon related work currently being explored with the mechanics of collaboration (Section 6.4).

6.1 Research goals and summary

My general research goal was to develop and validate a groupware evaluation methodology that is practical in terms of time, cost, logistics, and evaluator experience, while still identifying significant problems in a groupware system. This led directly to following specific research sub-goals:

1. Propose a new set of heuristics that can be used to detect usability problems in real-time, distributed groupware with a shared workspace.
2. Demonstrate that the adapted heuristic evaluation for groupware remains a 'discount' usability technique.

I have met these goals through the course of my research as detailed below.

Propose a new set of heuristics. In Chapter 2, I presented two complementary sets of groupware heuristics. The first set was developed from scratch based on the mechanics of collaboration framework (Gutwin and Greenberg 2000). In this case, I have adapted, restructured, and rephrased the framework as heuristics. In many situations, I have further augmented the theory with additional research not currently included in the framework. These heuristics are the focus of my research study. The second set is an extension of the Locales Framework heuristics that were first introduced by Greenberg et al. (1999). With these heuristics, I have further expanded each one and supplemented them with real-world examples to render then more understandable for individuals not intimately familiar with the Locales Framework. In both cases, the final results are groupware heuristics amenable to identifying teamwork usability problems in real-time, distributed groupware with a shared workspace.

Validating the heuristics. Demonstrating the methodology as a 'discount' usability technique enforces the notion that the heuristics are easy to learn and can subsequently be used by individuals that are not necessarily experts in groupware to identify usability problems. I explored this issue by having 27 inspectors with varying degrees of expertise in CSCW and no prior experience with the proposed heuristics evaluate two groupware systems. Results allowed me to derive with confidence the following general conclusions regarding the mechanics of collaboration heuristics (refer to Section 5.4 for more details):

1. They identify a category of teamwork usability problems inherent to the success of any real-time groupware system supporting a shared workspace.
2. A single evaluator will report a relatively low proportion of the total number of problems in a given interface.
3. Multiple inspectors will report more problems than a single individual.
4. A minimal amount of resources are required to identify an 'acceptable' proportion of problems in a given system.

Finally, a comparison with results reported by Nielsen (1992) with his traditional heuristic evaluation methodology shows that the proposed mechanics of collaboration heuristics obtains similar success. Unfortunately, an analysis of the Locales Framework heuristics does not demonstrate a similar level of success with their practicality.

6.2 Main contributions

The main contribution of this research study is its ability to help address a serious problem afflicting the acceptance of groupware: our inability to effectively and efficiently evaluate groupware usability and hence learn from these experiences. Given the scope of this problem, my research intention was not to solve it but rather explore an avenue for groupware evaluation that had not been seriously addressed by the CSCW community. In turn, I have made two contributions: I have confirmed a viable alternative to traditional groupware evaluation techniques and I have also introduced a practical set of heuristics as a starting point. I will now discuss these two contributions.

First, I have validated the assumption that we can adapt HCI evaluation techniques; specifically discount inspection methods, for the purposes of assessing groupware. Traditional work on groupware evaluation has focused on techniques in the areas of field studies and user observations. After all, field techniques provide the context necessary for 'accurate' evaluations and user observations exercise the system's use although in a controlled environment. Relatively speaking, minimal effort has been expended in the area of evaluating groupware with low-cost inspection methods such as Nielsen's heuristic evaluation. Hence, these types of methods have seen limited use in most industrial situations. In the field of HCI, it is accepted practice and common knowledge to mix and match evaluation techniques to provide the greatest coverage for identifying different types of usability problems. No single technique will uncover all problems in a given interface. Within this context, inspection methods play a role: to quickly and inexpensively identify usability bugs early in the design process in order to improve the system's usability. In the CSCW arena, we have not seen the same type of contribution from inspection methods. However, through this research study I demonstrate that it is possible to produce an inspection method for the meaningful, generalizable, and inexpensive evaluation of groupware.

Second, I have introduced and validated a set of groupware heuristics capable of identifying usability problems specific to real-time shared workspace groupware. Specifically, I used the mechanics of collaboration theory as the framework for these heuristics. However, these heuristics are still in their infancy. They are not intended to

address all existing categories of groupware systems nor all types of groupware usability problems. Despite these limitations, they still provide a realistic first-step for developing a viable technique for evaluating groupware usability.

This research has also produced a number of less important but still significant advancements:

- I have categorized problems related to the success of groupware.
- I have refined an existing set of heuristics based on the Locales Framework (Greenberg et al. 1999) to assist individuals unfamiliar with the framework conduct heuristic evaluations of groupware.
- I have further substantiated Gutwin's mechanics of collaboration and Fitzpatrick's Locales Framework as an approach to groupware evaluation.
- I have characterized a level of inspector performance and the results that can be expected when using the mechanics of collaboration heuristics to evaluate real-time, shared workspace groupware.

6.3 Further research

While addressing my research goals, I came across many interesting questions that I could not address because they were beyond the scope of my research. In this section I discuss three potential research areas that require further investigation.

Expanding the experiment. This research was a first attempt to produce heuristics for identifying usability bugs in real-time shared workspace groupware. As I look back on the experiment, I can identify areas were the experiment could be improved and other areas where it could be expanded. Since I have critiqued my methodology in Section 5.5.2, I do not propose to revisit this issue here. Rather I discuss studies that could be performed to extend and/or further validate the findings from this initial study:

1. Perform heuristic evaluations of other real-time shared workspace groupware, including systems earlier in the software engineering (SENG) development cycle. Nielsen (1992) averaged his findings over six separate case studies with half being paper prototypes and the other half running systems. This would help to further

substantiate the current results as well as analyze the heuristics' ability to identify problems early in the SENG process. Unfortunately, groupware systems require a minimal level of fidelity in order to effectively support the mechanics of collaboration. As a result, I suspect that the evaluation of systems with the groupware heuristics may be restricted to systems that are partially functional (i.e., medium fidelity) as opposed to paper prototypes (i.e., low fidelity). To compensate, the designer could use the heuristics to guide the early SENG design activities so that many potential problems can be addressed prior to formal evaluation activities taking place. In comparison to traditional HE, the groupware heuristics would have a more limited scope of evaluative applicability during the entire SENG process.

2. Recruit additional usability specialists with groupware expertise to perform heuristic evaluations with the proposed heuristics. This would help to further study the effect that expertise and experience in CSCW has on the inspector's performance. In addition, I could refine the experiment with tighter controls to further explore this relationship (see Section 5.5.2 for more details).
3. Conduct an empirical assessment of GroupDraw and Groove and compare the resulting usability problems with those discovered through groupware heuristic evaluations. This would help to characterize the scope, validity, and types of usability problems uncovered by the two sets of groupware heuristics. This would help to further address the issues presented in Section 5.5.1.1.
4. Evaluate systems that are not constrained to a shared workspace (e.g., Media Spaces, MUDs, MOOs) and others that are asynchronous (e.g., instant messengers, co-authoring tools). This would explore the ability of these heuristics to uncover usability problems in these types of systems. I suspect that the heuristics will have a limited amount of success since they were not explicitly designed for this groupware genre.

Evolving the heuristics. As outlined in section 2.3.2, Nielsen's original set of heuristics was revised based on contributions from other guidelines and knowledge gained from performing heuristic evaluations. I expect that the current groupware heuristics (especially the locales framework heuristics) will follow a similar evolution. One means

to revise the heuristics is to conduct a factor analysis similar to Nielsen's study (1994b). Nielsen took seven sets of usability principles and compared them with a database of existing usability problems in order to determine what heuristics explain actual usability problems. The result was a revised set of 10 heuristics. Performing a similar factor analysis would help to improve and/or further validate the existing groupware heuristics. However, this would involve two significant undertakings. First, a database of existing groupware usability problems, specific to real-time shared workspace groupware would have to be gathered. Unlike Nielsen's (1994b) database of 249 usability problems, there are few documented cases of groupware evaluation outlining specific usability problems that can be used to create a database. During my research, I noticed that the majority of existing groupware case studies focused on the issues surrounding their design. Less time was devoted to the evaluation of these systems and hence only a high-level cursory description of their assessment is presented. Second, one would have to find other sources of usability principles specific to groupware evaluation. Likewise, there are few published lists of usability principles.

With respect to the Locales Framework heuristics, more work can be done to improve them give the inability of the inspectors to effectively apply these heuristics. I think that the heuristics proposed in this research study provide a good beginning for future work.

Expanding the heuristics' area of coverage. Out of necessity, the heuristics brought forth by this research study were designed with a limited scope of application. Specifically, the mechanics of collaboration heuristics are aimed at uncovering problems tied to the low-level actions within a real-time shared workspace. Whereas, the Locales Framework heuristics address problems specific to the social factors influencing teamwork. However, there are a broad variety of factors (e.g., cultural, organizational, technical—see Chapter 1 for more details) that influence the successful adoption of groupware systems. The existing groupware heuristics do not provide complete coverage of all these factors. In their current state, they provide (as per my intentions) a means to heuristically evaluate a category of factors that influence the groupware usability of real-time systems. A more complete evaluation would involve applying complementary

usability evaluation techniques as well as heuristics with a wider area of coverage. To provide a comprehensive yet manageable set of heuristics to account for a reasonably large proportion of all potential problems, I would have to revise the current heuristics and/or create additional ones. However, this is a major undertaking given the complexity and variability of these external factors. Past research has explored the effects of social (e.g., Grudin 1994), organizational (e.g., Grudin 1988, Orlikowski 1992), and technical (e.g., Greenberg and Marwood 1994) factors on groupware acceptance. These may prove to be a useful starting point.

6.4 Related Work

In parallel to this study, the mechanics of collaboration heuristics are undergoing a complementary set of work. Specifically, Pinelle and Gutwin (2002) have developed a new groupware walkthrough methodology employing the mechanics of collaboration. The goal is to make use of contextual information when evaluating a system with this inspection methodology—something that is missing from a heuristic evaluation. In addition, Steves et al. (2001) have compared the results from performing a dual evaluation of a groupware system. One evaluation applied an inspection method with the raw mechanics of collaboration (as opposed to the revised heuristics found in this study) and the other through a user-based technique. They found the following:

1. Work context is often needed to evaluate groupware.
2. Discount usability inspection techniques do uncover a portion of the problems found by the end user interacting with the product in the actual work setting.
3. The two types of methodologies complement one another.

6.5 Conclusion

In this research study, my goals were to develop a set of heuristics for evaluating real-time shared workspace groupware that would leverage the advantages of Nielsen's existing heuristic evaluation methodology: cheap, intuitive, easy to motivate people to do it, does not require advance planning, can be used early in the development process, and

identifies usability problems significant to the success of an interface. As a result, I have created two sets of groupware heuristics (Chapter 3), develop a methodology for validating their practicality (Chapter 4), and demonstrated how the heuristics can be successfully used by inspectors with minimal expertise and training to identify usability problems in two groupware systems (Chapter 5). To that end, I have satisfied my research goals as described in Section 6.1. Consequently, the main contribution of this research study has been to assess and validate the feasibility of adapting an existing HCI discount usability technique for groupware evaluation. Finally, I have identified future research areas in Section 6.3 to further improve the existing heuristics and identify new sources of heuristics.

References

Benford, S., Bowers, J., Fahlen, L., Greenhalgh, C., and Snowdon, D. (1995). User Embodiment in Collaborative Virtual Environments. In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems,* v.1. pp.242-249.

Benford, S., Snowdon, D., Colebourne, A., O'Brien, J., and Rodden, T. (1997). Informing the design of collaborative virtual environments. In *Proceedings of the international ACM SIGGROUP Conference on Supporting group work.* pp.71-80.

Bentley, R., Hughes, J.A., Randall, D., Rodden, T., Sawyer, P., Shapiro, D. and Summerville, I. (1992). Ethnographically-informed systems design for air traffic control. In *Proceedings of the ACM CSCW '92 Conference on Computer-Supported Cooperative Work*. pp.123-129

Bias, R.G. (1994). The Pluralistic Usability Walkthrough. In Nielsen, J. and Mack, R.L. (Eds.), *Usability Inspection Methods.* John Wiley and Sons, New York. pp.63-76.

Bly, S. (1988). A Use of Drawing Surfaces in Different Collaborative Settings. In *Proceedings of ACM CSCW'88 Conference on Computer-Supported Cooperative Work*. pp.250-256.

Bly, S. and Minneman, S. (1990). Commune: A Shared Drawing Surface. In *Proceedings of ACM COIS'90*. pp.184-192.

Clark, H. (1996). *Using Language*. Cambridge University Press, Cambridge.

Clark, H. and Brennan, S. (1991). Grounding in Communication. In Baecker, R. (Ed.), *Readings in Groupware and Computer Supported Cooperative Work*. Morgan-Kaufman Publishers. pp.222-233.

Cockburn, A. and Greenberg, S. (1993). Making contact: Getting the group communicating with groupware. In *Proceedings of ACM COCS'93 Conference on Organizational Computing System.* pp.31-41.

Coutaz, J., Berard, F., Carraux, E., Astier, W., and Crowley, J.L. (1999). CoMedi: Using computer vision to support awareness and privacy in media spaces. In *Proceedings of ACM CHI'99 Conference on Human Factors in Computing Systems, Extended abstracts (Video demo)*. pp.13-14.

Cox, D. (1998). *Supporting Results Synthesis in Heuristic Evaluation***.** M.Sc. thesis, Department of Computer Science, University of Calgary, Calgary, Alberta.

Cugini, J., Damianos, L., Hirschman, L., Kozierok, R., Kurtz, J., Laskowski, S. and Scholtz, J. (1997). Methodology for Evaluation of Collaboration Systems. *The evaluation working group of the DARPA intelligent collaboration and visualization program, Rev. 3.0.* http://zing.ncsl.nist.gov/nist-icv/documents/method.html.

Damer, B., Kekenes, C., and Hoffman, T. (1996). Inhabited Digital Spaces. In *Proceedings of ACM CHI'96 Conference on Human Factors in Computing Systems*. pp.9-10.

Dix, A., Finlay, J., Abowd, G., and Beale, R. (1993). *Human-Computer Interaction,* Prentice Hall.

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

Dumas, J.S. and Redish, J.C. (1993). *A Practical Guide to Usability Testing.* Ablex Publishing.

Egido, C. (1990). Teleconferencing as a Technology to Support Cooperative Work: Its Possibilities and Limitations. In J. Galegher, R. Kraut, and C. Egido (Eds.), *Intellectual Teamwork: Social and Technological Foundations of Cooperative Work*. pp.351-372.

Ellis, C. and Gibbs, S. (1989). Concurrency Control in Groupware Systems. In *Proceedings of the ACM SIGMOD'89 International Conference on the Management of D*ata. pp.399-407.

Ellis, C., Gibbs, S., and Rein, G. (1991). Groupware: Some Issues and Experiences. *Communications of the ACM*, January, Vol. 34, No. 1. pp.9-28.

Ereback A.L. and Hook, K. (1994). Using Cognitive Walkthrough for Evaluating a CSCW Application. In *Proceedings of ACM CHI'94 Conference on Human Factors in Computing Systems*. pp.91-92.

Fish, R., Kraut, R., Root, R., and Rice, R. (1992). Evaluating Video as a Technology for Informal Communication. In *Proceedings of ACM CHI'92 Conference on Human Factors in Computing Systems.* pp.37-48.

Fitzpatrick, G., Mansfield, T., and Kaplan, S. (1996). Locales framework: Exploring foundations for collaboration support. In *Proceedings of the OzCHI '96 Sixth Australian Conference on Computer-Human Interaction*. pp.34-41.

Fitzpatrick, G. (1998). The Locales Framework: Understanding and Designing for Cooperative Work. PhD Thesis, Department of Computer Science and Electrical Engineering, The University of Queensland.

Furnas, G. (1986). Generalized fisheye views. In *Proceedings of ACM CHI'86 Conference on Human Factors in Computing Systems.* pp.18-23.

Fussell, S., Kraut, R., Lerch, F., Scherlis, W., McNally, M., and Cadiz, J. (1998). Coordination, Overload and Team Performance: Effects of Team Communication Strategies. In *Proceedings of ACM CSCW'98 Conference on Computer-Supported Cooperative Work*. pp.275-284.

Gaver, W. (1991). Sound Support for Collaboration, In *Proceedings of 2nd ECSCW'91 European Conference on Computer-Supported Cooperative Work.* pp.293-308.

Gaver, W. (1993). Synthesizing Auditory Icons. In *Proceedings of ACM INTERCHI'93 Conference on Human Factors and Computing Systems*. pp.228-235.

Gaver, W., Smith, R., and O'Shea, T. (1991). In *Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems.* pp.85-90.

Gaver, W., Moran, T., MacLean, A., Lovstrand, L., Dourish, P., Carter, K., and Buxton, W. (1992). Realizing a Video Environment: EuroPARC's RAVE System. In *Proceedings of ACM CHI'92 Conference on Human Factors in Computing Systems.* pp.27-35.

Goodwin, C. (1981). *Conversational Organization: Interaction Between Speakers and Hearers*, Academic Press.

Greenberg, S., Roseman, M., Webster, D., and Bohnet, R. (1992). Human and technical factors of distributed group drawing tools. *Interacting with Computers*, 4(1). pp.364-392.

Greenberg, S. and Marwood, D. (1994). Real time groupware as a distributed system: Concurrency control and its effect on the interface. In *Proceedings of the ACM CSCW'94 Conference on Computer-Supported Cooperative Work*. pp.207-217.

Greenberg, S., Gutwin, C. and Roseman, M. (1996). Semantic Telepointers for Groupware. In *Proceedings of OzCHI '96 Sixth Australian Conference on Computer-Human Interaction*.

Greenberg S. and Roseman, M. (1998). *Using a Room Metaphor to Ease Transitions in Groupware*. Research report 98/611/02, Dept of Computer Science, Univ. of Calgary, Alberta.

Greenberg, S., Fitzpatrick, G., Gutwin, C. and Kaplan, S. (1999). Adapting the Locales Framework for Heuristic Evaluation of Groupware. In *Proceedings of OZCHI'99 Australian Conference on Computer Human Interaction*. pp.28-30.

Greenberg, S. and Rounding, M. (2001). The Notification Collage: Posting Information to Public and Personal Displays. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* [CHI Letters 3(1)]. pp.515-521.

Grudin, J. (1988). Why CSCW Applications Fail: Problems in the Design and Evaluation of Organizational Interfaces. In *Proceedings of ACM CSCW '88 Conference on Computer-Supported Cooperative Work.* pp.85-93.

Grudin, J. (1994). Groupware and Social Dynamics: Eight Challenges for Developers. *Communications of the ACM.* January 1994, Volume 37, No. 1. pp.93-105.

Gutwin, C. (1997). Workspace Awareness in Real-Time Distributed Groupware. Ph.D Thesis, Dept of Computer Science, University of Calgary, Alberta.

Gutwin, C. and Greenberg, S. (1998). Design for Individuals, Design for Groups: Tradeoffs between power and workspace awareness. In *Proceedings of the ACM CSCW'98 Conference on Computer Supported Cooperative Work.* pp.207-216.

Gutwin, C. and Greenberg, S. (1999). The Effects of Workspace Awareness Support on the Usability of Real-Time Distributed Groupware. *ACM Transactions on Computer-Human Interaction,* 6(3). pp.243-281.

Gutwin, C. and Greenberg, S. (2000). The Mechanics of Collaboration: Developing Low Cost Usability Evaluation Methods for Shared Workspaces. *IEEE 9th Int'l Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET-ICE'00)*.

Heath, C. and Luff, P. (1991). Disembodied Conduct: Communication through Video in Multi-Media Office Environment. In *Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems.* pp.99-103.

Heath, C., Jirotka, M., Luff, P., and Hindmarsh, J. (1995). Unpacking Collaboration: The Interactional Organisation of Trading in a City Dealing Room. *Computer Supported Cooperative Work*, 3(2), pp.147-165.

Hill, W., Hollan, J., Wroblewski, D., and McCandless, T. (1992). Edit Wear and Read Wear. In *Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems.* pp.3-9.

Holtzblatt, K. and Beyer, H. (1996). Contextual Design: Principles and Practice. In Wixon, Dennis and Judith Ramey. *Field Methods Casebook for Software Design*. New York, NY: Wily Computer Publishing. pp.301-333.

Holtzblatt, K. and Beyer, H. (1999). Contextual Design. ACM Interactions 6, 1 (Jan. 1999). pp.32-42

Hughes, J., King, V., Rodden, T. and Andersen, H. (1994). Moving out from the control room: Ethnography in system design. In *Proceedings of ACM CSCW'94 Conference on Computer Supported Cooperative Work*. pp.429-439.

Isaacs, E. and Tang, J. (1993). What Video can and can't do for Collaboration: A Case Study. In *Proceedings of the First ACM International Conference on Multimedia*. pp. 199-206.

Ishii, H. (1990). TeamWorkStation: Towards a Seamless Shared Workspace. In *Proceedings of ACM CSCW'90 Conference on Computer Supported Cooperative Work*. pp.13-26.

Ishii, H. and Kobayashi, M. (1992). ClearBoard: A seamless medium for shared drawing and conversation with eye contact. In *Proceedings of the ACM CHI'92 Conference on Human Factors in Computing Systems*. pp.525-532.

Ishii, H., Kobayashi, M., and Grudin, J. (1992). Integration of interpersonal space and shared workspace: ClearBoard design and experiments. In *Proc ACM CSCW'92 Conference on Computer Supported Cooperative Work.* pp.33-42.

Jefferies, R., Miller, J., Wharton, C., and Uyeda, K. (1991). User interface evaluation in the real world: A comparison of four techniques. In *Proceedings of ACM CHI '91 Conference on Human Factors in Computing Systems*. pp.119-124.

Karat, C., Campbell, R., and Fiegel, T. (1992). Comparison of empirical testing and walkthrough methods in user interface evaluation. In *Proceedings of ACM CHI '92 Conference on Human Factors in Computing Systems*. pp.397-404.

Karat, H. (1994). Chapter 8: A Comparison of User Interface Evaluation Methods. In Nielsen, J. and Mack, R.L., (Eds.), *Usability Inspection Methods,* John Wiley and Sons, New York. pp.203-233.

Kintsch, W. (1988). The role of knowledge in discourse comprehension: A construction-integration model. *Psychological Review, 95.* pp.163-182.

Kraut, R., Egido, C., and Galegher, J. (1988). Patterns of Contact and Communication in Scientific Research Collaboration Remote Communications. In *Proceedings of ACM CSCW'88 Conference on Computer Supported Cooperative Work*. pp.1-12.

Lamping, J., Rao, R., and Pirolli, P. (1995). A Focus+Context Techniques Based on Hyperbolic Geometry for Visualizing Large Hierarchies. In *Proceedings of ACM CHI '95 Conference on Human Factors in Computing Systems*. pp.401-408.

Lauwers, J.C. and Lantz, K. (1990) Collaboration Awareness in Support of Collaboration Transparency: Requirements for the Next Generation of Shared Window Systems. In *Proceedings of ACM CSCW'90 Conference on Computer Supported Cooperative Work*. pp.303-311.

Lewis, C. and Rieman, J. (1994). Getting to Know Users and their Tasks. In Baecker R., Buxton W., Grudin J., and Greenberg S. (Eds.), *Readings in Human-Computer Interaction: Toward the Year 2000*, Morgan-Kaufmann. pp.122-127.

Mack, R. and Nielsen, J. (1994). Executive Summary. In Nielsen, J. and Mack, R. (Eds.), *Usability Inspection Methods*, John Wiley and Sons, New York. pp.1-23.

McGrath, J. (1996). Methodology matters: Doing research in the behavioural and social sciences. In Baecker, R., Grudin, J., Buxton, W., and Greenberg S. (Eds.), *Readings in Human Computer Interaction: Towards the Year 2000*, Morgan-Kaufmann. pp.152-169.

McLaughlin, M. (1984). *Conversation: How Talk is Organized*, Sage, Beverly Hills.

MIT Media Lab (1980). Discursions Video Disk.

Molich, R. and Nielsen J. (1990). Improving a human-computer dialogue. Communications of the ACM 33, 3 (March 1990), pp.338-348.

Muller, M.J., McClard, A., Bell, B., Dooley, S., Meiskey, L., Meskill, J.A., Sparks, R., and Tellam, D. (1995). Validating an extension to participatory heuristic evaluation: Quality of work and quality of work life. In *Proceeding of ACM CHI'95 Conference on Human Factors in Computing Systems,* [Conference Companion]. pp.115–116.

Munson, J. and Dewan, P. (1996). A Concurrency Control Framework for Collaborative Systems. In *Proceedings of ACM CSCW'96 Conference on Computer Supported Cooperative Work*. pp.278-287.

Nielsen, J. (1992). Finding usability problems through heuristic evaluation. In *Proc. of ACM CHI '92 Conference on Human Factors in Computing Systems.* pp.372-380.

Nielsen, J. (1993). *Usability Engineering*. Boston, Academic Press.

Nielsen, J. (1994a). Chapter 2: Heuristic Evaluation. In Nielsen, J. and Mack, R., (Eds.), *Usability Inspection Methods*, John Wiley and Sons, New York. pp.25-62.

Nielsen, J. (1994b). Enhancing the explanatory power of usability heuristics. In *Proc. of ACM CHI '94 Conference on Human Factors in Computing Systems.* pp.152-158.

Nielsen, J. (1995). Technology Transfer of Heuristic Evaluation and Usability Inspection. http://www.useit.com/papers/heuristic/learning_inspection.html

Nielsen, J. and Mack, R.L. (1994). *Usability Inspection Methods.* Eds. John Wiley and Sons, New York.

Nielsen, J. and Molich, R. (1990). Heuristic evaluation of user interfaces. In *Proceedings of ACM CHI '90 Conference on Human Factors in Computing Systems.* pp.249-256.

Norman, D.A. (1988). *The Psychology of Everyday Things.* Basic Books, New York.

Norman, D.A. (1993). *Things That Make Us Smart*, Addison-Wesley, Reading, Mass.

Orlikowski, W.J. (1992). Learning From Notes: Organizational Issues in Groupware Implementation. In *Proceedings of ACM CSCW '92 Conference on Computer Supported Cooperative Work*. pp.362-369.

Pinelle, D. (2000). A Survey of Groupware Evaluations in CSCW Proceedings. Research Report 00-1. Department of Computer Science. Univ. of Saskatoon, Sask.

Pinelle, D. and Gutwin, C. (2002). Groupware Walkthrough: Adding Context to Groupware Usability Evaluation. In *Proceedings of ACM CHI '02 Conference on Human Factors in Computing Systems.*

Polson, P.G. and Lewis, C.H. (1990). Theory based design for easily learned interfaces. *Human Computer Interaction, 5.* pp.191-220.

Randall, D. (1996). Ethnography and Systems Development: Bounding the Intersection (Parts 1 & 2). In *Proceedings of ACM CSCW '96 Conference on Computer Supported Cooperative Work*, [Tutorial notes].

Robinson, M. (1991). Computer-Supported Cooperative Work: Cases and Concepts In Baecker, R. (Eds.) *Readings in Groupware and Computer Supported Cooperative Work,* Morgan-Kaufman Publishers. pp.29-49.

Roseman, M. and Greenberg, S. (1996). Building Real Time Groupware with GroupKit, A Groupware Toolkit. March. *ACM Transactions on Computer Human Interaction*, 3(1), March. pp.66-106.

Rubin, J. (1994). *Handbook of usability testing : How to plan, design, and conduct effective tests.* Wiley, New York.

Sacks, H., Schegloff, E., and Jefferson, G. (1974). A Simplest Semantics for the Organization of Turn-Taking for Conversation, Language, 50. pp.696-735.

Salvador, T. and Mateas, M. (1997). Design Ethnography: Designing for Someone Else, Presentation notes.

Salvador, T., Scholtz, J., and Larson, J. (1996). The Denver Model of Groupware Design, *SIGCHI Bulletin*, 28(1). pp.52-58.

Sawyer, P., Flanders, A., and Wixon, D. (1996). Making a difference – The impact of inspections. In *Proceedings of ACM CHI '96 Conference on Human Factors in Computing Systems*. pp.376-382.

Segal, L. (1994). Effects of Checklist Interface on Non-Verbal Crew Communications, *NASA Ames Research Center*, Contractor Report 177639.

Shackel, B. (1990). Human Factors and Usability. In Preece, J. and Keller, L. (Eds.) *Human-Computer Interaction: Selected Readings*. Prentice Hall.

Short, J., Williams, E., and Christie, B. (1976a). Visual Communication and Social Interaction. In Baecker, R. (Eds.) *Readings in Groupware and Computer Supported Cooperative Work,* Morgan-Kaufman Publishers. pp.153-164.

Short, J., Williams, E., and Christie, B. (1976b). Communication Modes and Task Performance. In Baecker, R. (Eds.) *Readings in Groupware and Computer Supported Cooperative Work,* Morgan-Kaufman Publishers. pp.169-176.

Smith, R., O'Shea, T., O'Malley, C., Scanlon, E., and Taylor, J. (1989). Preliminary experiences with a distributed, multi-media, problem environment. In *Proceedings of EC-CSCW'89 European Conference on Computer Supported Cooperative Work*.

Stefik, M., Foster, G., Bobrow, D., Kahn, K., Lanning, S., and Suchman, L. (1987a). Beyond the Chalkboard: Computer Supported for Collaboration and Problem Solving in Meetings. *Communications of the ACM*, 30(1). pp.32-47.

Stefik, M., Bobrow, D., Foster, G., Lanning, S., and Tatar, D. (1987b). WYSIWIS Revised: Early Experiences with Multi-user Interfaces. *ACM Transactions on Office Information Systems*, 5(2). pp.147-167.

Steves, M.P., Morse, E., Gutwin, C. and Greenberg, S. (2001). A Comparison of Usage Evaluation and Inspection Methods for Assessing Groupware Usability. In *Proceedings of ACM Group'01,* ACM Press.

Tam, J. (2002). Supporting Change Awareness in Visual Workspaces. M.Sc. thesis, Department of Computer Science, University of Calgary, Alberta.

Tang, J. (1989). Listing, drawing, and gesturing in design: A study of the use of shared workspaces by design teams. PhD thesis, Department of Mechanical Engineering, Stanford University, California.

Tang, J. (1991). Findings from Observational Studies of Collaborative Work. *Intl J Man-Machine Studies*, 34(2). pp.143-160.

Tang, J. and Leifer, L. (1988). A Framework for Understanding the Workspace Activity of Design Teams. In *Proceedings of ACM CSCW'88 Conference on Computer Supported Cooperative Work.* pp.244-249.

Tang, J. and Minneman, S. (1990). VideowDraw: A Video Interface for Collaborative Drawing. In *Proceedings of ACM CHI'90 Conference on Human Factors in Computing Systems*. pp.313-320.

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

Tatar, D., Foster, G., and Bobrow, D. (1991). Design for Conversation: Lessons from Cognoter. *International Journal of Man-Machine Studies*, 34(2). pp.185-210.

Vertegaal, R. (1999). The GAZE Groupware System: Mediating Joint Attention in Multiparty Communication and Collaboration. In *Proceedings of ACM CHI'99 Conference on Human Factors in Computing Systems*. pp.294-301.

Wharton, C. (1994). Cognitive Walkthroughs: A Practitioners Guide. In Nielsen, J. and Mack, R.L. (Eds.), *Usability Inspection Methods.* John Wiley and Sons, New York. pp.105-140.

Appendix A: Heuristic Evaluation Training Materials

This appendix contains the following training materials that were used as part of the heuristic evaluation of the GroupDraw and Groove:

A.1. Consent form – completed by all evaluators prior to their participation in the study. Note: the text in square parentheses was omitted from the regular specialists' consent form.

A.2. Background information questionnaire – to assess the evaluators' knowledge and experience in the areas of HCI and CSCW.

A.3. Heuristic evaluation instructions – taken from a website, these instructions outline the evaluators' participation in the study, and how to install and run both groupware systems. Note: these instructions were given to the regular specialists. The novice evaluators' instructions were identical except all references to the Locales Framework heuristics were removed.

A.4. Pre-training feedback forms – to assess the ease with which all of the evaluators understood the mechanics of collaboration heuristics and the regular specialists understood the Locales Framework heuristics. Note: the same questionnaire was used to assess the evaluators' understanding of the heuristics after the training session.

A.5. Mechanics of Collaboration heuristics – used by all evaluators to inspect both groupware systems

A.6. Locales Framework heuristics – used by the regular specialists to inspect both groupware systems

A.7. Problem reports – used by the evaluators to record usability problems uncovered while inspecting both groupware systems with the two sets of heuristics.

A.1 Consent form

THE UNIVERSITY OF
CALGARY

Kevin Baker
Department of Computer Science, University of Calgary
Calgary, Alberta, CANADA T2N 1N4

CONSENT FORM

Research Project Title: Heuristic Evaluation of Groupware

Investigators: Kevin Baker and Saul Greenberg

This consent form is only part of the process of informed consent. It should give you the basic idea of what the research is about and what your participation will involve. If you would like more detail about something mentioned here, or information not included here, please ask. Please take the time to read this form carefully and to understand any accompanying information.

Experiment purpose:
The aim of this project is to evaluate the practicality of one set of heuristics for the evaluation of applications for groups working together through different computers (that is, groupware).

Participant recruitment and selection:
[This exercise is being given as part of your normal undergraduate training in human computer interaction and in evaluation techniques in particular.] To be recruited in this study, all you have to do is let us use and analyze the results of your exercise.

Procedure:
The entire study will require approximately 3 to 5 hours of your time over the span of two weeks, and will include the following activities:

1. Review of one or two sets of groupware heuristics.
2. Presentation of the groupware heuristics.
3. Evaluation of two groupware systems with one or two sets of groupware heuristics.

[**Confidentiality:**
Participant anonymity will be strictly maintained. Reports and presentations will refer to participants and groups using only an assigned number.]

Likelihood of discomfort:
There is no likelihood of discomfort or risk associated with participation.

Risks:
Your involvement or non-involvement in this study will in no way affect any class grade. The identity of participants and non-participants will be kept from the course instructor (Saul Greenberg).

Primary researchers:
Kevin Baker is a M.Sc. student in the department of Computer Science at the University of Calgary; this study is part of his research. His supervisor is Dr. Saul Greenberg, Professor in the department of Computer Science.

Your signature on this form indicates that you have understood to your satisfaction regarding participation in the research project and agree to participate as a participant. In no way does this waive your legal rights nor release the investigators or involved institutions from their legal and professional responsibilities. You are free to not answer specific items or questions on questionnaires. You are free to withdraw from the study at any time without penalty. Your continued participation should be as informed as your initial consent, so you should feel free to ask for clarification or new information throughout your participation. If you have further questions concerning matters related to this research, please contact:

Kevin Baker (kevbaker@nortelnetworks.com) or Dr. Saul Greenberg (saul@cpsc.ucalgary.ca)

---------- ----------
Participant Date

A.2 Background information questionnaire

QUESTIONNAIRE – BACKGROUND INFORMATION

The following questions are used to help ascertain your knowledge and experience in the areas of Human Computer Interaction (HCI) and Computer Supported Cooperative Work (CSCW).

Experience

If you are a student in Computer Science, what year are you in? _____

If you are a professional, what is your title? _____

If you are involved in UI design and/or evaluation, briefly describe what you do.

Human Computer Interaction questions

1. Have you ever received formal instruction in Human Computer Interaction (HCI)? (Check all that apply)

O	several courses	O	a module in a course
O	a full course	O	a class or two
O	a workshop	O	no training at all

2. Have you ever read any articles or books in HCI? (Check all that apply)

O 1. I regularly read articles and/or books on the topic
O 2. several books
O 3. a text book
O 4. a few popular articles
O 5. none

3. Have you ever designed a graphical user interface (GUI)? (Circle one)

many times a few times once never

4. Have you ever formally evaluated a user interface using an interface evaluation technique? (Circle one)

many times a few times once never

5. Have you ever formally evaluated a user interface using Nielsen's heuristic evaluation methodology? (Circle one)

many times a few times once never

Groupware and Computer Supported Cooperative Work questions

1. Have you ever received formal instruction in groupware and Computer Supported Cooperative Work (CSCW)? (Circle all that apply)

O	several courses	O	a module in a course
O	a full course	O	a class or two
O	a workshop	O	no training at all

2. Have you ever read any articles or books in groupware and CSCW? (Check all that apply)

O 1. I regularly read articles and/or books on the topic
O 2. several books
O 3. a text book
O 4. a few popular articles
O 5. none

3. Have you ever designed a groupware application? (Circle one)

many times a few times once never

4. Have you ever formally evaluated a groupware application? (Circle one)

many times a few times once never

A.3 Heuristic evaluation instructions

HEURISTIC EVALUATION OF GROUPWARE

background

Many organizations are becoming increasingly distributed. Work teams are often made up of people located at different sites, and experts are rarely present in the same location. As a result, the use of collaboration technology is fast becoming a necessity in these organizations, and the usability of the technology can have a large impact both on the overall efficiency and effectiveness of the team, and on the quality of the work that they do. Yet with the exception of a few systems, groupware is not widely used. One main reason is that these systems have serious usability problems, a situation caused in part by the lack of practical methodologies for evaluating them. In addition, most groupware systems are complex and introduce almost insurmountable obstacles to meaningful, generalizable and inexpensive evaluation. Without evaluation methods, groupware developers cannot learn from experience. Consequently, even today's collaborative systems contain usability problems that make them awkward to use.

objective of the study

We have yet to develop techniques to make groupware evaluation cost-effective within typical software project constraints. One way to address this dearth is to adapt evaluation techniques developed for single-user software usability. Within the field of human computer interaction (HCI), many low-cost evaluation techniques have moved out of the research arena and into accepted practice. Collectively, these become a toolkit. Each methodology highlights different usability issues and identifies different types of problems; therefore, evaluators can choose and mix appropriate techniques to fit the situation. We are adapting the heuristic evaluation inspection technique developed for conventional discount usability evaluation to groupware. In particular, we developed one set of heuristics from the Locales framework for group interaction, and another set based upon the low-level 'mechanics' of collaboration.

reason for your participation

To formally evaluate the groupware heuristics as a discount usability methodology, outside evaluators (that's you!) are needed to inspect and evaluate a real-time shared workspace application with these heuristics. The aim is to assess each evaluator's ability to learn and apply these heuristics.

your level of involvement

The table below outlines the steps involved with your participation in this research project. Note, if you cannot access any of the documents, please let me know ASAP and I can e-mail a copy to you.

step 1 - fill out consent form

Before beginning anything, I would appreciate that you print the attached consent form, review it, and sign it. This form highlights the terms of your participation in this research project. I will be picking up these forms during our 'training' session (refer to step 3).

Consent Form [PDF]

step 2 - read heuristics

Next, I would like you to read through the two sets of heuristics presented below.

Mechanics of Collaboration heuristics [HTML] or [PDF]

Locales Framework heuristics [HTML] or [PDF]

As you are reading each set of heuristics, I would ask that you provide feedback on each heuristic as per the following forms:

Feedback form for Mechanics of Collaboration heuristics [PDF] or [Word]

Feedback form for Locales Framework heuristics [PDF] or [Word]

I ask that you electronically fill out the feedback forms and e-mail them to me (kevbaker@nortelnetworks.com).

step 3 - attend a 'training' session

After you have read the material, we will be having a 'training' session. The objective of this session is to ensure that everybody understands the principles that form each heuristic. This will be done by reviewing each heuristic and using examples to illustrate their applicability to groupware. The session will be informal so you will have the opportunity to ask as many questions as you want.

step 4 - evaluate two groupware applications

The last step of your participation will involve your evaluation of two groupware applications, Groove and GroupKit, using the two sets of heuristics. The specific details regarding how to perform the evaluation and what to specifically evaluate will be discussed during our training session.
Here are the instructions for installing and running each application on your own computer:

If you want to install and run GroupKit, click here for details.

If you want to install and run Groove, click here for details.

I just want to thank you again for your participation in this research project. Your time and effort are greatly appreciated. If you have any questions, feel free to send an e-mail to me or Saul Greenberg.

Kevin Baker

INSTALLING AND RUNNING GROUPKIT

The following procedure describes how to install and run GroupKit on your workstation.

NOTE: You must install Tcl/Tk 8.0.4 on your computer before installing GroupKit.

installing GroupKit

step 1: downloading Tcl/Tk

1. go to: www.cpsc.ucalgary.ca/grouplab/developers/GroupKit/groupkit.html
2. within the 'Downloading Tcl/Tk' table, click the 'Complete distribution' link for Tcl/Tk version 8.0.4 to download the installation program to your computer. If you're using Microsoft Internet Explorer or America Online, you'll be asked if you wish to run or save the installation file. Choose Save this program to disk, then click OK. You'll be asked to specify a location on your computer to save the tcl804.exe file. Navigate to the desired location and click Save.
3. when the download has completed, double-click the Tcl/Tk executable
4. on the 'Welcome' window, click Next
5. on the 'Select Destination Directory' window, choose a location for the installation of the Tcl/Tk files and click Next
6. on the 'Select Installation Type' window, choose the 'Full installation' radio button and click Next
7. on the 'Ready to Install' window, click Next
8. on the 'Installation Complete!' window, click Finish

step 2: downloading GroupKit

1. go to: www.cpsc.ucalgary.ca/grouplab/developers/GroupKit/groupkit.html
2. within the 'Downloading GroupKit' table, click the 'Complete distribution' link that corresponds to your platform - this will install an executable on your computer If you're using Microsoft Internet Explorer or America Online, you'll be asked if you wish to run or save the installation file. Choose Save this program to disk, then click OK. You'll be asked to specify a location on your computer to save the gk51.exe file. Navigate to the desired location and click Save
3. when the download has completed, double-click the GroupKit executable
4. on the 'Do you wish to continue?' dialog box, click Yes
5. on the 'Welcome' window, click Next
6. on the 'Read me' window, click Next
7. on the 'Choose Destination Location' window, choose a location for the installation of the GroupKit files and click Next
8. on the 'Start Copying Files' window, click Next
9. on the 'Setup Complete' window, click Finish

running GroupKit

Under GroupKit, you don't start up programs directly, but do it through a registration system. Without getting into the underlying philosophy, here's the steps you should go through to get something going:

step 1: running the registrar

Choose a machine to run a server process called the registrar. To run the registrar on this machine, double-click the "registrar.tcl" file found at the root of the GroupKit distribution (normally located at C:\Program Files\GroupKit\registrar.tcl).

Note: the registrar does not create a window when it is running. This process is just running in the background.

step 2: opening a session manager

Start up the open registration session manager, by double-clicking on "openreg.tcl" (normally located at C:\Program Files\GroupKit\openreg.tcl). The first time you run, GroupKit will ask for some information about you and your environment and then create a preferences file for you. Specifically, you will be prompted to enter your name, your internet domain and the name of the machine that is running the registrar (as per step 1 above).

Next, a dialog box labeled 'Connect to where?' will pop up. The two text fields will be pre-filled with name of the machine running the registrar and the port (note: Port 9357 is the default). Click Connect to begin a session.

The 'Open Registration' window will now appear. The "Conferences" pane on the left shows the names of any running conferences. Selecting one will show who is in the conference in the "Participants" pane on the right.

starting a GroupKit conference

1. To create a new conference, pull down the Conferences menu. A list of conferences will be shown. Choose one, give it a new name if you want to in the pop-up dialog box, and click "Create". For example, create a "Simple Sketchpad" conference. The program should appear on your screen.

2. To pretend you were a different user wanting to join the conference. Create another session manager on your computer (refer to step 2 of 'running GroupKit'). You should see the name of the Simple Sketchpad conference in the conferences list. If you click it, you will get the list of active users (just one). To join the conference, double click on the conference name, and you will get another copy of the GroupDraw application up on your screen.

INSTALLING AND RUNNING GROOVE

The following procedure describes how to install Groove from the Web site. First, you'll have to register, and then download the installation program. Finally, you'll be able to install Groove and create your Groove account. Before beginning the installation, make sure your Internet connection is active, and close all open applications.

installing Groove

1. go to: http://components.groove.net/thankyou.html
2. you'll be prompted to register with Groove Networks' Web site. Complete all fields and click continue
3. once you're registered, you'll see important information related to installing Groove, such as late-breaking news and system requirements. Click continue to go to the Download Groove page and proceed with the installation. Click Download self-extracting EXE to download the installation program. If you're using Microsoft Internet Explorer or America Online, you'll be asked if you wish to run or save the installation file. Choose Save this program to disk, then click OK. You'll be asked to specify a location on your computer to save the GrooveSetup.exe file. Navigate to the desired location and click Save.
4. when the download has completed, double-click the GrooveSetup.exe file. You'll be asked a series of questions, such as where to install the application and which components to install. Click Next each time to install Groove with the most common options. When the installation is complete, make sure that View the README file and Launch Groove are selected, then click Finish.
5. If you're asked to reboot your computer when the installation completes, reboot.

running Groove

1. Launch Groove from the Start menu or by double-clicking the desktop icon.
2. When you launch Groove, you'll have to create a Groove account to use the application. It's important to choose a name for your Groove account that will make you recognizable to your contacts. Your passphrase should be easy to remember, yet difficult for anyone else to guess. Once you've created your account, you'll use this username and passphrase whenever you launch the Groove application.

A.4 Pre-training feedback form

FEEDBACK – MECHANICS OF COLLABORATION HEURISTICS

The following question asks you to consider the ease with which you were able to understand each of the heuristics derived from the mechanics of collaboration. Space has been provided so that you may include any additional comments. This information will be used to help:

1) identify any 'problem' spots with each heuristic and
2) assess the ability of individuals to comprehend each heuristic without any prior training.

Heuristic 1: Provide the means for intentional and appropriate verbal communication

Do you feel you have a good understanding of this heuristic? (place an X beside your choice)

_ 1. Not at all: I just don't understand the idea.
_ 2. I understand the general point, but am unclear about the details
_ 3. I understand in general the details of the heuristic, but I don't believe I can use it to analyze an interface
_ 4. I understand the heuristic and I feel I can use it to analyze an interface

Comments:

Heuristic 2: Provide the means for intentional and appropriate gestural communication

Do you feel you have a good understanding of this heuristic? (place an X beside your choice)

_ 1. Not at all: I just don't understand the idea.
_ 2. I understand the general point, but am unclear about the details
_ 3. I understand in general the details of the heuristic, but I don't believe I can use it to analyze an interface
_ 4. I understand the heuristic and I feel I can use it to analyze an interface

Comments:

Heuristic 3: Provide consequential communication of an individual's embodiment

Do you feel you have a good understanding of this heuristic? (place an X beside your choice)

_ 1. Not at all: I just don't understand the idea.

_ 2. I understand the general point, but am unclear about the details

_ 3. I understand in general the details of the heuristic, but I don't believe I can use it to analyze an interface

_ 4. I understand the heuristic and I feel I can use it to analyze an interface

Comments:

Heuristic 4: Provide consequential communication of shared artifacts

Do you feel you have a good understanding of this heuristic? (place an X beside your choice)

_ 1. Not at all: I just don't understand the idea.

_ 2. I understand the general point, but am unclear about the details

_ 3. I understand in general the details of the heuristic, but I don't believe I can use it to analyze an interface

_ 4. I understand the heuristic and I feel I can use it to analyze an interface

Comments:

Heuristic 5: Provide protection

Do you feel you have a good understanding of this heuristic? (place an X beside your choice)

_ 1. Not at all: I just don't understand the idea.

_ 2. I understand the general point, but am unclear about the details

_ 3. I understand in general the details of the heuristic, but I don't believe I can use it to analyze an interface

_ 4. I understand the heuristic and I feel I can use it to analyze an interface

Comments:

Heuristic 6: Manage the transitions between tightly and loosely-couple collaboration

Do you feel you have a good understanding of this heuristic? (place an X beside your choice)

_ 1. Not at all: I just don't understand the idea.

_ 2. I understand the general point, but am unclear about the details

_ 3. I understand in general the details of the heuristic, but I don't believe I can use it to analyze an interface

_ 4. I understand the heuristic and I feel I can use it to analyze an interface

Comments:

Heuristic 7: Support people with the coordination of their actions

Do you feel you have a good understanding of this heuristic? (place an X beside your choice)

_ 1. Not at all: I just don't understand the idea.

_ 2. I understand the general point, but am unclear about the details

_ 3. I understand in general the details of the heuristic, but I don't believe I can use it to analyze an interface

_ 4. I understand the heuristic and I feel I can use it to analyze an interface

Comments:

Heuristic 8: Facilitate finding collaborators and establishing contact

Do you feel you have a good understanding of this heuristic? (place an X beside your choice)

_ 1. Not at all: I just don't understand the idea.

_ 2. I understand the general point, but am unclear about the details

_ 3. I understand in general the details of the heuristic, but I don't believe I can use it to analyze an interface

_ 4. I understand the heuristic and I feel I can use it to analyze an interface

Comments:

FEEDBACK – LOCALES FRAMEWORK HEURISTICS

The following question asks you to consider the ease with which you were able to understand each of the heuristics that were derived from the Locales Framework. Space has been provided so that you may include any additional comments. This information will be used to help:

1) identify any 'problem' spots with each heuristic and
2) assess the ability of individuals to comprehend each heuristic without any prior training.

Heuristic 1: Provide centers (locales)

Do you feel you have a good understanding of this heuristic? (place an X beside your choice)

_ 1. Not at all: I just don't understand the idea.
_ 2. I understand the general point, but am unclear about the details
_ 3. I understand in general the details of the heuristic, but I don't believe I can use it to analyze an interface
_ 4. I understand the heuristic and I feel I can use it to analyze an interface

Comments:

Heuristic 2: Provide mutuality within locales

Do you feel you have a good understanding of this heuristic? (place an X beside your choice)

_ 1. Not at all: I just don't understand the idea.
_ 2. I understand the general point, but am unclear about the details
_ 3. I understand in general the details of the heuristic, but I don't believe I can use it to analyze an interface
_ 4. I understand the heuristic and I feel I can use it to analyze an interface

Comments:

Heuristic 3: Allow individual views

Do you feel you have a good understanding of this heuristic? (place an X beside your choice)

_ 1. Not at all: I just don't understand the idea.

_ 2. I understand the general point, but am unclear about the details

_ 3. I understand in general the details of the heuristic, but I don't believe I can use it to analyze an interface

_ 4. I understand the heuristic and I feel I can use it to analyze an interface

Comments:

Heuristic 4: Allow people to manage and stay aware of their evolving interactions over time

Do you feel you have a good understanding of this heuristic? (place an X beside your choice)

_ 1. Not at all: I just don't understand the idea.

_ 2. I understand the general point, but am unclear about the details

_ 3. I understand in general the details of the heuristic, but I don't believe I can use it to analyze an interface

_ 4. I understand the heuristic and I feel I can use it to analyze an interface

Comments:

Heuristic 5: Provide a way to organize and relate locales to one another

Do you feel you have a good understanding of this heuristic? (place an X beside your choice)

_ 1. Not at all: I just don't understand the idea.

_ 2. I understand the general point, but am unclear about the details

_ 3. I understand in general the details of the heuristic, but I don't believe I can use it to analyze an interface

_ 4. I understand the heuristic and I feel I can use it to analyze an interface

Comments:

A.5 Mechanics of Collaboration heuristics

HEURISTICS FOR SUPPORTING THE MECHANICS OF COLLABORATION

Overview

The following set of eight heuristics are aimed to help inspectors evaluate how a *shared workspace* groupware system supports (or fails to support) the ability of distance-separated people to communicate and collaborate with artifacts through a visual medium. A shared workspace is a bounded space where people can see and manipulate artifacts related to their activities. Groupware applications that provide a shared workspace include shared editors, group drawing programs, multiplayer games, and distributed control systems. These heuristics are also tailored for workspaces that are used by groups of approximately five people.

The basis for this set of heuristics is the *mechanics of collaboration* framework. These mechanics were developed from an extensive analysis of shared workspace usage and theory. It describes the low level actions and interactions that small groups of people do if they are to complete a task effectively. Basic actions include communication, coordination, planning, monitoring, assistance, and protection. The underlying idea of the framework is that while some usability problems in groupware systems are strongly tied to social or organizational issues in which the system has been deployed, others are a result of poor support for the basic activities of collaborative work in shared spaces. It is these basic activities that the framework articulates.

Heuristic 1: Provide the means for intentional and appropriate verbal communication

Theory. In face-to-face settings, the prevalent form of communication between group members is verbal conversations. These conversations are *intentional* as group members consciously exchange information with each other. We use this information to establish a common understanding of the task at hand. Three ways we pick up information from verbal exchanges are:

- People may talk explicitly about what they are doing and where they are working within a shared workspace. These discussions typically take place when an individual asks a direct question such as "What are you doing?" or when the group is discussing the division of labour.
- People can gather information by overhearing others' conversations. Although a conversation between two people may not explicitly include a third, people understand that the exchange of information is public and therefore freely available to others.
- People can listen to the running commentary that others tend to produce in conjunction with their actions. That is, people may say what they are doing as they are doing it. This may seem as if people are talking to themselves, but they are really doing it to inform others of their actions. This "verbal shadowing" provides additional information without forcing people to explicitly enter into a conversation.

What this means for groupware. Most real-time groupware does not support intentional verbal communications directly. They assume that any communication channel (text, audio, video) is supplied via another means (e.g. a telephone). Whether the system supports a channel for intentional communication or relies on another tool, collaborators must be able to effectively communicate with one another. As we will see, verbal exchanges also play a prominent role in satisfying the criteria of many of the other heuristics.

Techniques used in groupware. Depending on the task, verbal conversations can be facilitated via a digital audio link or text chat facility between participants. These can be implemented in many ways, and each has consequences on how well the communication activity is supported. Text chat, for example, can be sent in 'chunks' (e.g. type a line and send) or real-time (e.g. character by character). Text can be useful for short or

sporadic interactions, or where it is impractical to provide an audio connection. However, text is limited. It is cumbersome to carry on long discussions and extremely difficult to type a running commentary of one's actions. Text channels are typically directed at others, so it also may be difficult for a third person to 'overhear' a conversation. In practice, lengthy or highly interactive meetings require an audio channel, typically supplied by a telephone. Digital audio is available in some systems, but currently lacks quality due to bandwidth and latency problems.

Video can also support verbal conversations. However, there are questions concerning the benefits of adding video to an audio channel. People have found it difficult to engage in negotiation over distributed media since video can introduce distractions that interfere with the accuracy of interpersonal evaluations. Plus, when conversations shift to a meta-level, participants typically turn away from the monitor displaying the virtual workspace and communicate across the video monitor. Video quality is also notoriously poor in most computers. Images can be blurry and small, and latency can be large. When the frame rate is compromised, images are jumpy and do not really inform others about conversational nuances. However, video is useful for managing the mechanics of conversations since the visual actions accompanying the talk can be seen (this will be discussed in more detail in later heuristics). In this capacity, video may be useful for handling higher-level conversations that are independent of the workspace. While video has appeal and utility, the designer cannot assume that including video is the answer for meeting all intentional communication requirements. A stand-alone audio channel may suffice if the majority of the interaction is performed in conjunction with the shared workspace and its artifacts.

Heuristic 2: Provide the means for intentional and appropriate gestural communication

Theory. Explicit gestures and other visual actions are also used alongside verbal exchanges to carry out intentional communication. These actions are a crucial resource. Group members use gestures to reference objects within the workspace, express ideas, mediate the group's interactions, focus the attention of others, and demonstrate actions. In addition, they communicate information that cannot be readily expressed otherwise. As a result, gestures play a prominent role in all work surface activity for design teams collaborating over whiteboards and paper on tabletops (around 35% of all actions).

These gestures are *intentional.* People use them to directly support the conversation and convey task information. They are not the incidental gestures that can accompany human dialog such as scratching your head or tapping your fingers. These types of gestures are discussed in the next heuristic.

Intentional gestural communication can take many forms.

- *Illustration* occurs when speech is acted out or emphasized with gestures. For example, people typically express distances by showing a gap between their thumb and index finger.
- *Emblems* occur when gestures replace words such as a nod or shake of the head to indicate 'yes' or 'no'.
- *Deictic reference* or *deixis* happens when people reference objects in the workspace with a combination of intentional gestures and communication. A common activity is pointing to an object and saying "this one".

What this means for groupware. Because groupware participants are distance-separated, intentional gestures are invisible unless they are supported directly by the system. However, in many groupware applications, the representation of bodies is still completely absent. Consequently, gestural communications are lost. As a minimum, groupware must support intentional gestural communication (e.g. illustration, emblem, deixis) by making all gestures visible to all group members. In addition, the system must maintain the association between the gesture and its author. Maximizing the information gathered from gestures in a shared virtual workspace can go a long way in helping people maintain awareness of others and their actions.

Gestures in a virtual workspace can occur via a simple movement of the mouse. As a result, these actions sometimes go unnoticed by other group members, even when the system supports them. This is especially true when our attention is focused elsewhere or the collaboration involves multiple participants interacting simultaneously. Groupware must ensure that gestures are not easily missed while not being overly distracting.

Gestures are often performed in relation to objects in the workspace and to the location of others in the group. This happens when we point to another person or an object in the workspace (e.g. deixis). We use this spatial relationship between gestures and their referents to help interpret collaborative activities. Therefore, groupware needs to not only convey the gestures, but to display them in relation to the objects or people they are referencing.

As in the case of illustration and deixis, gestures are generally coupled with verbal talk. To ensure that each gesture remains tightly coordinated with its verbal exchange, both the audio connection and the gestures need to be shared among all participants without delays. Another key element is the ability to associate both the gesture and verbal exchange with the specific person.

Techniques used in groupware. In groupware, supporting gestures is typically done (if at all) via some form of *embodiment.* Embodiment involves representing each user to others and to themselves within a shared workspace. Common techniques for conveying embodiment include telepointers, avatars, and video images.

Telepointers are the simplest means for supporting embodiment in a virtual workspace. A person's cursor, made visible to all, allows one to gesture and point to objects in the workspace. Specifically, telepointers can facilitate gestural actions via the following means:

- Each person has their own large and uniquely identifiable telepointer that can be used simultaneously with the others.
- Telepointers are always visible within the work surface by all participants.
- They appear with no apparent delay in order to remain synchronized with verbal exchanges.
- They maintain their same relative location to the worksurface objects across each user's display.

While telepointers are limited 2D caricatures of the rich gestures people do with their hands, they are a huge improvement over nothing at all.

Avatars are synthetic bodies representing the people who populate a 2D or 3D landscape. While most avatars are extremely crude, some transmit limited hand and body gestures. The idea is to capture real world gestures and have them appear in the simulated space. This technique is especially powerful if this capture is based on normal, implicit actions, rather than having someone explicitly control the actions of the avatar.

Video can also recreate real bodies within virtual workspaces. Conventional video windows or monitors are not sufficient because gestures are detached from workspace objects. A different approach is to mix and fuse the video of the person, their hands, and the worksurface into a single image. When done correctly, the final image comprises both the artifacts and the person, where gestures relative to the artifacts are maintained.

Heuristic 3: Provide consequential communication of an individual's embodiment

Theory. A person's body interacting with a physical workspace is an immediate and continuous source of awareness information. In these settings, our bodily actions (such as position, posture, and movements of head, arms, hands, and eyes) *unintentionally* give off information, which is picked up by others. The mechanism of seeing and hearing other people active in the workspace is called *consequential communication*. Watching other people work is our primary means for understanding what's going on, who else is present in the workspace, their whereabouts, and what they are doing This information is fundamental for generating and sustaining teamwork.

Consequential bodily communication is not intentional in the same manner as explicit gestures (heuristic 2). In this case, an individual does not consciously perform the actions to inform others—these are incidental actions that we naturally produce during our interactions. In addition, the perceiver simply gathers what is available from the other.

Unintentional body language can be divided into two categories: *actions coupled with the workspace* and *actions coupled to the conversation*.

Actions coupled with the workspace include such activities as gaze awareness (i.e. knowing where another person is looking), seeing a participant move towards an object or artifact, and hearing characteristic sounds as people go about their activities.

Actions coupled to conversation are the subtle cues (e.g. facial expressions, voice intonation) we pick up from our conversational partners that allow us to continually adapt and adjust our verbal conduct. Although these cues may not occur within the workspace, they are essential for providing *conversational awareness.* This awareness helps people maintain a sense of what is happening in a conversation. In turn, we can mediate turn-taking, focus attention, detect and repair conversational breakdowns, and build a common ground of joint knowledge and activities. For example, a speaker will use eye contact to assess the listener's attentiveness by starting an utterance, waiting for eye contact from the listener, and then re-starting the utterance.

What this means for groupware. The goal of supporting consequential communication in real-time groupware is to capture and transmit both the explicit and subtle dynamics that occur between collaborating participants. To support those actions coupled with the workspace, the system needs to present the following information in a timely manner:

- Location – where each group member is active in the shared space. This is closely related to the notion of gaze awareness.
- Gaze and View – where the person is looking and what they can see.
- Action – what each individual is doing
- Presence – to tell 'at a glance' if others are present in the workspace
- Identity – to convey the identity of each individual. For instance, the system can help distinguish between different individuals even if you don't know who they are. Or, once you have learned someone's identity, you might be able to recognize him or her again.

To support those actions coupled with the conversation, the subtle visual and verbal cues must be captured and transmitted by the groupware application. Cues may be visual, like facial expressions, body language (e.g., head nods), eye contact, and gestures emphasizing talk. Or they may be verbal such as intonation, inflection, pauses, back channels, and the use of particular words.

Techniques used in groupware. The embodiment techniques previously discussed (heuristic 2) are a start for supporting visual cues coupled with workspace activities. For example, telepointers allow us to see people moving towards an object, which helps to predict their actions. They can also change their shape to reflect a natural action such as pointing or writing. Telepointers may hint at where its owner is looking, although there is no guarantee that the person is really doing so. Avatars can go one step further by linking the 'gaze direction' of the avatar to the point of view. This signals its owner's approximate field of view in the environment. However, these embodiment techniques are very limited. While they do help, the poor match of these embodiments to a person's actual body movements means that there are many consequential gestures that are not captured and transmitted. Video systems that mix a person's video embodiment into the workspace are more successful for capturing visual and verbal cues coupled with the workspace activities.

CSCW research on conversational awareness has concentrated on how technological mediums affect distributed conversations and how to substitute enough richness in synthetic *video* and *audio* channels to reach an acceptable level of awareness.

Video. Special care must be taken with camera placement to ensure eye contact and gaze awareness are accurate. In most desktop systems, we see speakers 'looking' at our navels or hairline simply because cameras are mounted on top or underneath the monitor. The bandwidth must be able to accommodate full video. The use of compressed video results in small, jerky and often blurred images that loose many of the subtle body cues. Even with full video, zooming the camera in to capture facial expressions ('talking head' view) means that other body gestures are not visible. Yet zooming out to include the whole body compromises image fidelity and resolution. Unfortunately, video is still quite limited in this capacity. Despite these implementation problems, many signals can still be read through a video channel. For instance, facial expressions are still apparent even on poor quality video or small monitors. A video channel is also useful for enhancing verbal descriptions and managing extended pauses.

Audio. Audio is also a concern for consequential communication. The voice channel must be of good audio quality so that the clarity of a person's speech dynamics is not compromised. A full-duplex channel enables people to speak at the same time. In turn, listeners can interrupt or inject back-channel utterances such as 'ok' and 'ums'. The effectiveness of an audio channel is also sensitive to delays. Even small delays (in excess of 150-200 ms) can disrupt participants' ability to reach mutual understanding and reduce their satisfaction with the conversation. Unfortunately, when the voice is non-directional, people find it difficult to associate a voice with a particular speaker (e.g. multi-point teleconferencing). Speaker identification and turn-taking is difficult when a teleconference involves four or more people.

Heuristic 4: Provide consequential communication of shared artifacts

Theory. In face-to-face settings, consequential communication also involves information *unintentionally* given off by artifacts as individuals manipulate them. This information is called *feedback* when it informs the person who is manipulating the artifact, and *feedthrough* when it informs others who are watching. Physical artifacts naturally provide visual and acoustic feedback and feedthrough. Visually, these artifacts depict their state in their physical representation as well as form spatial relationships with one another. In addition, an artifact's appearance sometimes shows traces of its history or wear—gradual and unavoidable change due to use—explaining how it has transformed into its current state (e.g., object wear). Acoustically, physical artifacts make characteristic sounds as they are manipulated (e.g. scratch of a pencil on paper). Seeing and/or hearing an artifact as it is being handled helps to determine what others are doing to it and how it has changed.

Another resource available in face-to-face interactions is the ability to identify the person manipulating the artifact. Knowing who produced the action provides context for making sense of this action, and helps collaborators mediate their interactions. Actions within a shared workspace are often used to bid for turn-taking in a conversation. Therefore, being able to associate the action with the initiator helps others yield their turn.

What this means for groupware. Due to the spatial separation between artifact and actor, feedthrough tends to be the only vehicle for sharing artifact information between groupware participants. However, groupware complicates feedthrough since it limits the expressivity of artifacts and actions. For instance, direct manipulation of artifacts in a virtual workspace (e.g., dragging and dropping a small object) is not as visible or distinguishable as its face-to-face equivalents. As a result, these actions can easily go unnoticed and be harder to differentiate from visually similar events. They can also transpire instantaneously (e.g., a click of a button), leaving minimal warning of their occurrence and minimal time to see and understand them. At the lowest level, the shared virtual workspace must immediately transmit the local user's feedback to all remote users. This involves not just showing the final position of a moved object, but also its selection and the intermediate steps of its move. Presenting this interim feedback ensures changes within the workspace do not happen instantaneously. It also ensures there is no reduction in the information other people can gather about the activity while it is happening.

In contrast to the physical world, interactions within a virtual workspace are not limited to direct manipulation. Symbolic commands and indirect manipulation afford users the ability to interact with artifacts in ways that are often not possible in the real world. These are shortcuts that stress quick execution while providing minimal feedback. Examples of symbolic manipulation techniques are menu commands,

buttons, keyboard shortcuts, and toolbars. Unfortunately, indirect manipulation of a virtual object has no workspace representation at all. Unless specifically designed into the interface, we receive little information about the author of the action, its occurrence, or its progress. Virtual artifacts must be designed to be more expressive to ensure that this information can be gathered as they are manipulated.

The spatial separation between actor and artifact means that the two are not always coupled in the virtual workspace. Whether it is direct or indirect manipulation of an artifact, all collaborators must be able to identify the producer of the action. Identifying the producer helps to provide context to the action.

In contrast to physical artifacts, virtual ones also do not naturally produce characteristic sounds; therefore, many groupware workspaces are silent and lacking this form of consequential communication. In order to hear sounds in groupware, designers must create and add synthetic replacements to virtual artifacts and events. Adding sound to a virtual workspace is a useful addition, especially when screen space is limited or if the information is of a peripheral nature. Currently, it is difficult determining the best sound to correspond with each action, recreating the subtlety and range of naturally occurring workspace sounds, and finding sounds that work effectively in combination so that each may be heard and understood. In addition, the directional and proximal components of sounds tend to be weak since workspaces are 2D with limited audio output devices.

Physical objects typically display temporal information regarding how they were created and how they have changed over time through use. In contrast, virtual objects have a consistent appearance and manipulating them does not inherently leave evidence of wear. Their monotony provides fewer clues for maintaining *change awareness* of the artifact—the ability to track what has happened to the object and what another person has been doing. Groupware designers must overcome this monotony to ensure that people do not have fewer signs for determining what has happened and what another person has been doing.

Techniques used in groupware. In groupware, feedback during the direct manipulation of an object can be transmitted to all participants via *action feedthrough*. In this way each individual sees the initial, intermittent, and final state of an artifact as it is manipulated. Early groupware systems imposed "what you see is what I see" (WYSIWIS) view sharing where all participants saw the exact same actions as they occurred in the workspace. As a result, everybody shares a common view of the workspace. *Process feedthrough* ensures that local feedback of an indirect manipulation is also transmitted to all others to help them determine what is about to happen. Intermediate states of indirect manipulation can be presented using visual techniques such as action indicators and animation. In either case, the identity of the individual manipulating the artifact can be transmitted via the embodiment techniques described in heuristics 2.

Sound is a means to support feedthrough. Adding sound to an event or action provides details regarding its occurrence, type, location, and duration even when it happens in a part of the workspace that is not visible. For instance, our desktop employs sounds (at the user's discretion) to signify events such as an incoming e-mail. The same techniques can be (and have been) applied to groupware applications.

Techniques for displaying the history of virtual artifacts include 'edit wear' and 'read wear'. In this capacity, the designer can attach temporal information to a workspace object. In turn, the object could show the amount of time since it was last manipulated, who was the last person to change it, or how often it has been changed recently. For instance, each time an object is used, its workspace representation can become darker.

Heuristic 5: Provide Protection

Theory. Collaboration over a shared physical workspace often involves multiple individuals active in the space. Along with interacting simultaneously within the shared space, participants may want to work within the same part of the space at the same time. To this extent, concurrent activity in the workspace can be a valuable resource. Collaborators can work in parallel and it helps the group smoothly negotiate the use of the workspace. The competition for conversational turn-taking can also be reduced as one person works in the shared space while another talks and holds the audio floor. All of this is fairly easy to do in a face-to-

face setting. We can easily see what and where others are working and can in turn mediate our interactions accordingly.

What this means for groupware. Groupware users must be able to concurrently interact with the shared virtual workspace. Concurrent access should be allowed to the point of being able to work in the same area at the same time. In face-to-face settings, physical constraints regulate this type of activity. It is hard for people to interfere with one another without physically getting in each other's way. However, these constraints are not present in groupware. Collaborators can act in parallel within the same space and simultaneously manipulate shared objects. Although beneficial, this can introduce the potential for conflict. People can inadvertently interfere with work that others are doing now, or alter or destroy work that others have done. To regulate concurrent access within a virtual workspace, groupware applications require protocols to provide protection in attempt to minimize conflicts.

Protocols are mutually agreed upon ways of interacting. Within groupware, these protocols may be left to the control of the participants, *social protocols* or built into the system, *technological protocols*.

Social protocols. Social scientists have found that people naturally follow *social protocols* for mediating their interactions. This includes turn-taking in conversations and the ways shared physical objects are managed. People also learn to anticipate each other's actions and take action based on their expectations or predictions of what others will do in the future. With these resources, participants can avoid conflicting actions. If conflicts do occur, people are quite capable of repairing their negative effects and consider it part of their natural dialog. Under these circumstances, groupware must provide the means to correct the results from conflicts (e.g. undo functionality).

As a result, many groupware systems give all collaborators equal rights to all objects. To provide protection, they rely on people's natural abilities to anticipate actions, mediate events and resolve conflicting interactions. Under these circumstances, the system's role can be limited to providing a good sense of what is going on in the shared workspace. This involves transmitting awareness of others' actions and feedback of shared objects. To anticipate events, collaborators must be able to keep an eye on their own work, noticing what effects others' actions could have and taking actions to prevent certain kinds of activity.

Technological protocols. To assist with social protocols, *technological protocols* can be designed into the system. This ensures that the correct protocol is followed, provides structure to the group's activity, and assists less experienced users. Specific techniques will be discussed shortly.

There has been a considerable amount of debate between using social or technological protocols for providing protection. Advocates of social protocols argue that we don't fully understand the way that people work together. Consequently, we cannot effectively design a protocol describing work. Technological protocols may not match a group's peculiar working style or they may constrain a group that needs to use different processes for different activities. Natural protocols are also constantly evolving and changing throughout a session. While humans can quickly and fluidly adapt to these changes, protocols embedded in the system may find it more difficult to transition smoothly. Although social protocols will generally work, this approach may not always be acceptable. By allowing conflicts to occur systems force the users to resolve these events after they have been detected. This is undesirable if the result is lost work. Users may prefer 'prevention' to 'cure'. The quality of awareness will also not function well with high-latency communications where there is a delay in delivering one user's actions to others. Technological protocols help to alleviate these problems.

There is a delicate balance between technical and social protocols. Poor support of other heuristics may result in people not having enough information to effectively use social protocols. Yet over-reliance on technical protocols may produce a system that gets in the way of what people are trying to do. The ultimate choice depends on the groupware activities. In either case, users must be able to concurrently access the shared workspace while being afforded the capabilities to be protected from others' conflicting actions.

Techniques used in groupware. To effectively use social protocols to mediate interactions, groupware participants require the ability to talk with one another. An audio channel would suffice (as per heuristic 1). A combination of the awareness techniques described in the previous heuristics is also helpful for keeping

an eye on the actions of others. For example, remote handles can graphically warn users that someone else is already using an item.

To provide protection, technical measures such as access control, concurrency control, undo, version control, and turn-taking can been implemented. For example, concurrency control could manage conflicting actions and thus guard against inconsistencies. However, concurrency control in groupware must be handled differently than traditional database methods since the user is an active part of the process. People performing highly interactive activities may not tolerate delays introduced by conservative locking and serialization schemes. Access control can also be used to determine who can access a groupware object and when. Access control may be desirable when people wish to have their own private objects that only they can manipulate and/or view. Within groupware access control must be managed in a lightweight, fine-grained fashion. If not, it will be intrusive: people will fight with the system as they move between available and protected objects.

Heuristic 6: Manage the transitions between tightly and loosely-coupled collaboration

Theory. A shared physical workspace has a dual private/public nature. For instance, individuals can begin working on an idea privately in some portion of the workspace. As the idea matures, it can be gradually made public to everyone sharing a common view of the workspace. The dual nature of the workspace enables individuals to work independently or jointly with others.

In the context of collaboration, *coupling* is the degree to which people are working together. In general terms, coupling is the amount of work that one person can do before they require discussion, instruction, information, or consultation with another. People continually shift back and forth between *loosely- and tightly-coupled collaboration* where they move fluidly between individual and group work. This typically occurs when people need to plan subsequent activities, see an opportunity to collaborate, or have reached a point in their task that necessitates another's contribution. For example, assisting others with their tasks is a vital part of collaboration whereby individuals move from loose to tight coupling.

What this means for groupware. Groupware systems must maintain the dual nature of the workspace so that people can choose between private and public work. Therefore, each individual must have the ability to navigate the workspace independently to see and manipulate the objects they need. As a result, a person's view can be the same as everybody else or completely different.

Unfortunately, when people can look at different areas of the entire workspace, events that occur outside their viewport are not visible. Therefore, an overhead associated with independent workspace navigation is the ability to manage the transitions between loose and tight coupling. To manage these transitions, the groupware designer must provide enough awareness so that they can be done smoothly. For instance, assistance may be opportunistic and informal. This situation makes it easy for one person to help another without a prior request. Under these circumstances, groupware must provide awareness of others to help people determine when assistance is required and what is appropriate. Therefore, people should be able to focus their attention on different parts of the workspace when they are doing individual work. Assistance may also be explicitly requested. Individuals will make indirect statements indicating the need for assistance and their partner leaves their tasks to help out, and then return to what they were doing. To explicitly ask for assistance, the group members require a means to intentionally communicate with one another (as per heuristic 1). In any situation, to move from loose to tight coupling you need to know the context of others' actions (i.e. what they are doing, their goals, the stage of their task, and the state of their work area).

Maintaining awareness of others when we are not working in the same area of the workspace is further exacerbated because display devices are smaller with lower resolution when compared with the normal human field of view. The reduction in size forces people to work through a small viewport, thus only a small part of a large workspace is visible at any given time. The reduction in resolution makes it harder to see and distinguish artifacts from one another; therefore, visual events can be more difficult to perceive,

differentiate, and comprehend. Groupware must account for the display's limitations when maintaining awareness outside the individual's viewport.

To support the flow of work when the group members are not working simultaneously, there is a need for a way to convey the progress or changes made to the object since last time. There also has to be a smooth coordination between joint work and individual work. For example, private work may involve transferring a shared document, or pieces of it, to a personal workstation. Conversely, public work would involve transferring the changed document back and merging it in a graceful fashion.

Techniques used in groupware. Implementing WYSIWIS ensures that people stay aware of one another's activities since everybody sees the exact same actions as they occur in the workspace (i.e. used to support artifact feedthrough as per heuristic 4). However, this approach is often too restrictive when people regularly move back and forth between individual and shared work. More recent systems allow people to move and change their viewports independently. As a result, people can view the workspace objects that interest them. This is called relaxed-WYSIWIS view sharing. To manage transitions, groupware uses visual techniques that situate awareness information in the workspace. However, these techniques encounter the same visibility problem: the relevant part of the workspace has to be visible for the techniques to be of any use. When the workspace is too large to fit into a single window, the entire area outside the local user's viewport cannot be seen. Special techniques are required to make the relevant parts of the workspace visible. Examples are included here.

Overviews provide a birds-eye view of the entire workspace in a small secondary window. A properly designed overview makes embodiments, activities, and feedthrough visible, regardless of where they take place in the workspace. As a result, users see both their local view as well as everybody else's whereabouts in the workspace and the general structure of their activities. An overview displaying additional awareness information through telepointers and *view rectangles* (an outline showing what another can see) is called a *radar view*. With radar views, people can easily pursue individual work by moving their view rectangle in the radar view to a different part of the workspace. Conversely, if they want to work closely together, they can quickly align their view rectangles atop one another. The cost is the extra screen real estate occupied by the radar window and the cognitive distraction of having to refer to it.

Detail views duplicate a selected part of the shared workspace in a secondary viewport in order to provide a closer look at another person's view of the workspace. In comparison to overviews, they show a smaller portion of the workspace but it is larger with increased resolution (e.g., a cursor's-eye view duplicates the other person's work area and actions in its full size). As a result, one can gather more detailed information regarding the other person's activities.

Focus+context views provide both local detail and global context within the same display, usually through information visualization techniques such as fisheye views or special lenses. Analogous to a magnifying glass, they enlarge an area of interest while maintaining its context within the greater workspace.

Heuristic 7: Support people with the coordination of their actions

Theory. An integral part of face-to-face collaboration is how group members mediate their interactions by taking turns and negotiating the sharing of the common workspace. People organize their actions in a shared workspace to help avoid conflicts with others and efficiently complete the task at hand. Coordinating actions involves making some tasks happen in the right order, at the right time while meeting the task's constraints. The effectiveness of communication and collaboration can be enhanced if the group's activities are coordinated. Symptoms of poor coordination include people bumping into one another, duplication of actions, or multiple individuals trying to concurrently access shared resources.

Coordination of action is a higher order activity, a necessary overhead when several parties are performing a task. It is built upon many mechanisms listed in the previous heuristics. For instance:

- Explicit communication (heuristic 1) is used to accomplish such tasks as discussing how work is to be performed.

- Gestures and other forms of visual conduct help coordinate work tasks. Specifically, people use explicit gestures (heuristic 2) to direct attention by referring to existing workspace objects. Similarly, visual cues (heuristic 3) such as facial expressions help mediate conversational turn taking.
- Coordination is also accomplished by the way objects are shared during the work process. In addition, the way information is generated and organized in the shared workspace acts as a focus that curbs digressions and keeps the group working in a coordinated fashion. Both events use workspace awareness (e.g., heuristics 4 and 6) to inform participants about the temporal and spatial boundaries of others' actions.
- The workspace itself plays a key role in mediating interactions between collaborators, primarily as a means to focus the group's attention. Collaborators can also occupy their own attention by working privately as well as draw personal attention prior to enacting a gesture. The close physical proximity in these situations helps mediate actions since collaborators are peripherally aware of each other.

All mechanisms are beneficial for different levels of coordination. At the fine-grained level, awareness is evident in continuous actions where people are working with shared objects. One example is the way that people manage to avoid making contact with one another while collaborating within a confined space. In these settings, the close physical proximity among the collaborators provides peripheral awareness of each other and any actions. Thus collaborators can fit the next action into the correct sequence. On a larger scale, groups regularly reorganize the division of labour (i.e., what each person will do next as the task progresses). These decisions depend on what the others are doing and have done, what they are still going to do, and what is left to do in the task. Knowing activities and locations can help determine who should do what task next.

The coordination of activities at both levels is also assisted by anticipation. People take action based on their expectations or predictions of what others will do in the future. At the fine-grained level, people predict events by projecting forward from the immediate past. If you see someone reaching for a pen, you might predict that they are going to grab it. Based on this prediction you can take action (e.g. pick up the pen and hand it to the other person or alter your own movements to avoid a collision). In this case, anticipation is supported by the up-to-the-moment knowledge of the activity (i.e. where the other person's hand is moving) and the location (i.e. the location of the hand in relation to the pen). In addition, your prediction could have taken into account other knowledge, such as the other person's current activities and if they require a pen. When prediction happens on a larger scale, people learn which elements of situations are repeated and constant. People are experts at recognizing patterns in events, and quickly begin to predict what will come next in situations that they have been in before.

What this means for groupware. People are generally skilled at coordinating their communication and collaboration with each other. Consequently, groupware applications should not impose a structure that attempts to manage the interactions for them. Instead, tools should facilitate the participants' own abilities to coordinate their communication and collaboration. Workspace awareness helps people determine whether others' current workspace events match the patterns that they have learned. Therefore, groupware must allow individuals to remain aware of others within a shared workspace and the nature of their actions. In addition, collaborators must be able to see all actions within the context of the entire workspace even when people are working in different parts of it. Collaborators must also have the ability to communicate with one another. It is the inclusion of all these supports within groupware systems that enables people to effectively coordinate their activities at both a fine-grain level or on a larger scale.

It is important to realize that this heuristic bundles up many previous heuristics. In order to use it one has to 'step back' from the other low level mechanics and ask how all these work together to help people coordinate what they do. That is, it may be possible to satisfy individual heuristics, but people may still have trouble coordinating their actions because the means for satisfying individual heuristics don't work together very well.

Techniques used in groupware. Techniques for supporting coordination within a shared workspace have been presented in the previous heuristics. Verbal communication can be supported as per heuristic 1. The visual techniques presented in heuristics 2 through 5 will help to establish awareness of others and their actions within the workspace. Implementing relaxed WYSIWIS helps to ensure that this awareness is maintained even when the collaborators are not sharing a common view of the workspace (heuristic 6).

Heuristic 8: Facilitate finding collaborators and establishing contact

Theory. One problem with groupware is that it is not clear how people actually begin their groupware meetings. In everyday life, relatively few meetings are *formal* i.e., scheduled in advance with pre-arranged participants. These are usually arranged via e-mail, telephone, formal meeting requests, etc. In reality most meetings are *informal* encounters: unscheduled, spontaneous or one-person initiated. These are facilitated by physical proximity since co-located individuals can maintain awareness of who is around. Under these circumstances, people frequently come in contact with one another through casual interactions (e.g. people bump into each other in hallways) and are able to initiate and conduct conversations with little effort. While conversations may not be lengthy, much can occur: people coordinate actions, exchange information, or offer opportunities. Successful teams rely on regular, informal, and unplanned contact between their members.

What this means for groupware. It is more difficult to support informal groupware encounters since the bottleneck to rich spontaneous interactions is distance. In electronic communities, people are distributed. Therefore designers need to support how the group determines who is around and their availability if they are to initiate contact in a real-time groupware session. Even when potential collaborators have been identified, many mundane factors now interfere with making contact over computers. People must know electronic addresses and select from many communication channels and applications that are available to the group. People must ready software, equipment, and each other well in advance for real-time remote conferencing. From a technical perspective, workstations may not have the same software or support the necessary media (e.g., digital audio). In addition, specialized equipment may not be available (e.g., video cameras), poor networks may limit interactions, and applications must run across platforms.

Groupware applications must overcome the distance barrier that inhibits informal encounters. Information on potential collaborators must be provided so that they can be easily found and their availability for group work can be determined. To make this possible, we must be able to make ourselves available (i.e. provide a presence to help others gauge our availability). If collaborators are able and willing to engage in a groupware session, we must be able to initiate contact with minimal effort. This can be as easy as a single click or providing the necessary information (e.g. phone number or e-mail address).

Techniques used in groupware. Instant messaging provides a simple but limited mechanism for seeing who is around and establishing contact with them. A more comprehensive approach uses a room metaphor whereby people know who is around and can easily move into conversation and work. For instance, the following means can be used to facilitate informal interactions:

Being available. People can pursue single user activities in a room. Analogous to a physical room used for both individual and group activities, people will be around more often and thus available for real time encounters.

Knowing who is around and available for interaction. Within a spatial setting, we sense who else is around as we walk down hallways, glance into offices, and see others in public spaces. We judge their availability by a variety of cues such as if their door is open and how busy they look. A room metaphor provides a similar sense of presence and awareness by displaying the inhabitants of each room within the electronic community (via a user list) as well as status information. To judge availability, four ‘door states’ can indicate a person’s desire to be interrupted. As in real life, a wide and partially open door icon indicates a willingness to accept interruptions, while a barred door suggests that the room and its inhabitants are inaccessible. Although the door metaphor is a familiar concept, it is somewhat heavy-weight since it requires an explicit action on the occupant’s part to set the door state.

Establishing contact. There are several ways of establishing contact with individuals. One can just enter a populated room, and they are immediately connected to others. Room occupants also see the person enter because their picture appears. To avoid people 'popping up' out of nowhere in a room, cues can be used to warn the current occupants of a new person's approach to the room before they enter. A room-specific chat facility allows conversations to occur. If a person would rather initiate a conversation before entering a room, they can page somebody. Phone calls can be established with phone numbers from each person's business card. To establish contact with others not currently logged on, a person can leave a note in a room suggesting a meeting time and place.

Working together. The power of the room metaphor is that, once in a room, the working context is immediately available. All tools and room artifacts are at hand and new tools can be easily added.

Mechanics of Collaboration Heuristics

Provide the means for intentional and appropriate verbal communication

The prevalent form of communication between group members is verbal conversations. This allows a common understanding of the task at hand to be established. Support verbal exchanges or a viable alternative.

Provide the mean for intentional and appropriate gestural communication

Allow explicit gestures and other visual actions to be visible since they are done in direct support of the conversation and help convey task information. Support illustration, emblem and deixis.

Provide consequential communication of an individual's embodiment

A person's body interacting with a computational workspace must unintentionally give off information to others. This is the primary mechanism for maintaining awareness and sustaining teamwork. Unintentional body language must be coupled with both the workspace and its artifacts, and the conversation.

Provide consequential communication of shared artifacts (i.e. artifact feedthrough)

Make artifacts expressive so they give off information as they are manipulated. Support artifact feedthrough.

Provide protection

The user should be protected from inadvertently interfering with work that others are doing now, or altering or destroying work that they have done. Provide mechanisms to support social protocols and/or implement technical means to ensure protection.

Manage the transitions between tightly and loosely-coupled collaboration

Users should be able to focus their attention on different parts of the workspace when they are doing individual work in order to maintain awareness of others. Provide techniques for making relevant parts of the workspace visible.

Support people with the coordination of their actions

Support awareness of others' activities to ensure people can coordinate their actions in order to avoid conflicts and make tasks happen in the correct sequence.

Facilitate finding collaborators and establishing contact

Information on potential collaborators must be provided so that they can be easily found and their availability for group work can be determined. Initiation of contact should be possible with minimal effort.

A.6 Locales Framework heuristics

HEURISTICS FOR SUPPORTING THE LOCALES FRAMEWORK

Overview

The Locales Framework is an approach to help people understand the nature of social activity and teamwork, and how to support these activities. The motivation is to help inform the design of groupware. More formally, the Locales Framework comprises the following five interdependent aspects: 1) Locales foundations, 2) Mutuality, 3) Individual view over multiple locales, 4) Interaction trajectories, and 5) Civic structures. These five aspects have been recast as heuristics that ask whether a groupware's interface allows certain social phenomena to occur. These five heuristics along with the supporting theory are presented below. The Locales Framework deals with the social issues surrounding the use of groupware. These heuristics are therefore better suited for evaluating comprehensive collaborative environments and how they co-exist with a group's everyday methods for communicating and collaborating.

The Locales Framework was not developed as a usability evaluation method; however, it seems amenable to this for several reasons. First, this is a general framework (as opposed to specialized) for understanding the fundamental aspects of teamwork. It also describes a small set of different but inter-dependent perspectives of the characteristics of teamwork, where each could be reconsidered as a heuristic. Second, it has been validated as both a way to understand existing work practices, and to motivate the design of new systems. Thus it is reasonable to expect that it can be used as a standard against which to evaluate groupware.

HEURISTIC 1: Provide centers (locales)

Theory. Within the Locales Framework, a *social world* is defined as a group of people with a common purpose. This includes a product team designing a software application or a social club organizing a weekend camping trip. In this capacity, each individual naturally belongs to multiple social worlds. Social worlds need *site* and *means* to facilitate their shared interactions. For this purpose, social world members construct *locales* as a center to gather the site and means as needed.

As a *site*, a locale is a conceptual place or domain whereby social world members come together to collaborate. A place is defined by its members. A place is not confined to the physical configuration of a particular space. Consequently, a place can be physical, such as meeting room or virtual, such as a groupware application (e.g. NetMeetings). While the latter is not realized in physical space, it still provides the site for the social world to collaborate. A site can also be a locale for multiple social worlds. For example, one meeting room can be a different sort of locale for different groups.

Each site is furnished with the *means* by which people collaborate and communicate. Means include:

- work objects or objects as the focus of the social world activities (e.g. a shared drawing surface);
- tool objects used to create or modify other objects (e.g. pens, scissors, computers); and
- communication objects used to facilitate work-related tasks (e.g. telephone, e-mail).

People use a variety of tools, objects, resources, etc. to construct locales for the purposes of getting their shared work done. Our everyday locale, for example, includes mechanisms from the physical domain such as a computer, telephone, meeting rooms, printers, paper, and pen. As a complement, we also use mechanisms from the virtual domain such as e-mail, spreadsheets, Internet, and word processors. In most instances, our daily activities require little effort to bring together all of these mechanisms to accomplish shared work. For instance, we are quite adept at using the telephone and e-mail to help get our work done. How we choose to do the shared work depends on personal preference and the perceived strengths, weaknesses, and best uses for each mechanism.

Means also include the *created* and *inherent* infrastructure of the space itself. The created infrastructure includes the doors, windows, walls, etc., in a physical space and the network partitions and file types in a virtual space. The inherent infrastructure of the space encompasses the presence of air, and the laws of sound and light transmission in physical spaces, and the absence of sensory support in a virtual space. In this capacity, the means can play a role in facilitating access and supporting privacy and security within a locale. One example is the use of the office door by members (in both physical and virtual locales) to control access to their own spaces.

As a whole, a locale is dynamic in its composition. The site and means will vary in accordance with the changing demands of the social world members as they pursue their central goal. For instance, meetings can take place in different rooms on different days. Depending on the situation, communication will also switch between face-to-face, via e-mail or over the telephone.

What this means for groupware. There are situations where collaboration is cumbersome since all the mechanisms needed to accomplish the shared activity cannot be (or easily) brought together. This is typically the case when the members of a social world are distributed. For example, while it is easy to communicate with others via the telephone, it is hard for us to bring a visual workspace into our conversation. In this instance and others, the virtual domain lends itself to the development of software tools for the support of group work. To help alleviate the problem in our example, we can introduce a groupware application that supports a shared workspace. However, this alone is not sufficient. The application must interact with the other tools we use in the locale. In other words, any new groupware tool that we introduce into a locale must be integrated with the everyday methods we use to communicate and collaborate. To illustrate with our example, the new groupware application should ideally tie in with the telephone to facilitate discussing the visual workspace during our conversation. In this capacity, a single groupware application does not need to encompass all of the sites and means necessary to support a social world's interactions. Locales can be constructed by providing interoperability between many individual tools such as Instant messaging, electronic whiteboards, email, and so on. This enables people to effortlessly switch between each tool as their needs change. Consequently, if this collection of tools is quite fragmented, it may be heavyweight for people to construct a locale, or to return to it later on.

Another approach to providing locales involves the designer constructing a complete virtual locale furnished with all the appropriate site and means. This is the 'all-in-one' solution. At the lowest level, this involves providing a site that brings people together so that they are in the same place at the same time even though this may not be physically possible. Members of a social world must also be able to communicate with one another. This can be accomplished via some means as a chat tool, audio link, or video channel. In addition, the groupware system must support the members with the functionality to perform the task at hand. For example, individuals collaborating over a drawing require a shared workspace and a suite of tools. Virtual locales should also be dynamic so they can evolve along with the people, the artifacts, and the purposes that define them. This involves providing the capability to add (and remove) tools, applications, objects, etc., as the social world members require them. A common approach to constructing a complete virtual locale has been to extend our notions of the physical world to their design. For instance, some systems employ a room metaphor to manage distributed multi-user interactions.

Ultimately, the design of any virtual locale will depend upon a number of issues. The designer must take into account the social worlds involved, their resource requirements, their members, and how these relate to the shared goal. In addition, the social organizations within the world must be identified along with the range of possibilities for the social world processes. These processes include the definition of members, the negotiation of roles and responsibilities, privacy and access, and the joining and leaving of groups.

HEURISTIC 2: Provide mutuality within locales

Theory. Mutuality helps to facilitate interactions within locales and maintain a sense of shared place and shared work. Mutuality concerns the mechanisms available in the domain to support the projection of *presence* information and receipt of *awareness* information. Both are essential for effective collaboration.

Presence. Interactants require some way of representing or making themselves *present* in the locale. Presence information can include identity, form, and function. As well, it can include possibilities for interaction, current activity, current state, etc. as appropriate for the interaction.

Awareness. For communication and interaction purposes, interactants need to maintain *awareness* of others' presence, the artifacts comprising the locale, where things are located, and how things are changing. This information enables the receiver to answer various questions about others – who, what, when, where, why and how.

Both presence and awareness are facilitated by various mechanisms that are part of the locale. These mechanisms may be inherent to the domain itself, such as air for the transmission of sound waves in the physical domain, or they may be incorporated on behalf of the designer, such as artifact feedthrough. The ability to make use of mechanisms to support the exchange of presence-awareness information is mediated by *capability* and *choice*.

Capability has to do with the individual's ability to utilize the available mechanisms. For instance, bandwidth constraints might determine whether presence is projected across a network via text, audio, video or a combination of these mediums. Awareness capabilities for humans are largely to do with perception via our senses.

The *choice* of which mechanisms to use in a particular interaction can depend on what form of presence-awareness is most appropriate given the context in which the interaction is to take place and/or personal preference. If I want to communicate with somebody in another city, I can send them a letter or e-mail, call them, or visit them. Each provides different mechanisms for presenting presence/awareness, with different temporal and cost implications.

What this means for groupware. Unlike the physical domain, there are few if any inherent mechanisms available in the virtual domain to support mutuality. In the physical domain, the embodied nature of actions and the physicality of individuals and artifacts naturally provide presence and awareness. Therefore, little or no effort is required on the part of individuals to maintain mutuality in face-to-face collaboration. However, in a virtual domain, geometric space and corporeality have no meaning. For instance, virtual actions are not coupled to the producer nor do they make intrinsic sounds. In addition, virtual objects do not have a tangible form. As such, projecting presence and maintaining awareness within a virtual environment is inherently difficult.

Mechanisms to capture and display presence-awareness information within virtual locales must be explicitly developed by the groupware designer. The type of presence-awareness that must be supported will naturally depend on the goal of the system (which ultimately is related to the needs of the social world). For this reason, we can categorize awareness into *workspace*, *informal*, and *conversational*. Each type will have a different influence on the system design. Since presence is a necessary pre-condition for awareness, the type of presence that a system must support will also vary with each type.

Workspace awareness (WA) is the up-to-the-moment understanding of another person's interaction with a shared workspace. This involves knowledge about where someone is working, what they are doing, and what they are going to do next. This information is useful for many of the activities of collaboration—for coordinating action, managing coupling, talking about the task, anticipating others' actions, and finding opportunities to assist one another. In general, WA must be supported for any type of groupware system that has a shared virtual workspace such as a whiteboard or document.

Informal awareness involves knowing who's currently around, whether they're available or busy, and what sort of activity they're engaged in. People need informal awareness in order to find opportunities for collaboration and people to collaborate with. This type of presence-awareness is critical for media spaces, virtual communities, and instant messaging systems such as ICQ and Messenger. In these situations, members of the social world need to keep track of who is around and their availability for collaboration.

Conversational awareness helps people maintain a sense of what is happening in a conversation. The subtle cues (e.g. facial expressions, voice intonation) we pick up from our conversational partners help us

continually adjust our verbal behaviour. In turn, we can mediate turn-taking, focus attention, detect and repair conversational breakdown, and build a common ground of joint knowledge and activities. Real-time communication within a virtual locale is only effective when conversational awareness is supported. To support this type of awareness, the subtle visual and verbal cues must be captured and transmitted by the groupware application. Visual cues include facial expressions, body language (e.g. head nods), eye contact, or gestures emphasizing talk. Whereas, verbal cues include intonation, pauses, or the use of particular words.

The format for providing presence-awareness information in a virtual locale will also depend on the synchronicity of the system. In asynchronous use, the workspace presents past activity information, so as to give an individual awareness of the activities of other participants integrated with the work object itself. Whereas, synchronous participation by a number of people in a group locale might require that all have a high degree of presence and awareness for that locale (i.e. presenting information as it happens).

HEURISTIC 3: Allow individual views

Theory. Despite pursuing a common goal, a social world is not a homogeneous group. It is made up of *individuals* who each bring their own perspective and goals to the group. Therefore, individuals within a social world will rarely see the same things all the time in exactly the same way. In fact, each individual will have their own distinct view of the locale reflecting their current level of involvement. This level of involvement will often depend on each person's current responsibilities, roles, activities, needs, or interests. Furthermore, the individual's view of their locale will vary in accordance with their changing level of involvement in the social world.

Individuals typically belong to multiple social worlds. They simultaneously interact with each social world with varying degrees of focus and participation in the necessary locales. Therefore, the individual must continually manage their level of involvement in multiple locales. As a result, each individual neither works with a fixed group view of each locale nor attends to each locale in turn. Conversely, the individual will often work with a view composed of multiple locales – their *workaday world*. To get their tasks done, individuals will draw into their *workaday view* the things they need as they need them from the relevant locales. The individual will continually negotiate and manage their view over multiple locales as their needs change. The individual is often able to move relatively seamlessly between these locale views with their different activities while maintaining varying levels of focus and participation in the other locales. For example, I can be composing a document on my computer when the phone will ring and I am suddenly discussing plans for the weekend with a friend.

What this means for groupware. Each member of a social world must be able to view an individual locale or an aggregate of multiple locales from their own perspective. In each case, the view should relate to their responsibilities, activities, and interests and should reflect their degree of focus and participation in each locale. Within a single locale, each person must be able to define and view the portion of the shared work that interests them. In the case of a shared document, multiple individuals may need to work on it simultaneously with each person responsible for a different section. This requires each person to have their own distinct view of the document (i.e. their own section). Another scenario is our ability within an instant messaging system such as Messenger to construct our own personal contact lists.

As a member of multiple social worlds, people should be able to define which objects from which locales they want to see at the current time. This entails the ability for users to dynamically vary their view in accordance with different levels of interaction. The user should also be able to transition smoothly and easily from one locale to another as needed. The activities and artifacts from multiple social worlds can potentially be visible simultaneously in one screen. For example, our desktop includes objects drawn from activities from different locales – part of our view over those locales. There might be a chapter from my research project, but there might also be a work presentation that I must review, an e-mail from a friend living in Boston, and so on. Switching between these tasks is switching between different social world involvements and their different locales. Seamless and fluid transitions between different social world activities are supported by the minimal, subtle effort required to move a cursor or to type a command.

There is little conscious and physical effort expended to change social world contexts or to move to a different locale.

HEURISTIC 4: Allow people to manage and stay aware of their evolving interactions over time

Theory. Interactions within a social world happen over time. The *interaction trajectory* captures all of the aspects of these interactions that occur within and across locales as they evolve and change over time. It is about the social world in action in its locale(s), and the co-evolution of action, locale, and social world as the trajectory unfolds. Interaction trajectory can be described in terms of its *past*, *present*, and *future*.

- The *past* trajectory is its history, the course of action that has happened up until the current point in time.
- The *present* of trajectory is concerned with the current actions as they occurring. Some issues of interest include: What is the vision that frames this action? What are the conditions under which this action will be taken? What are the possible actions that could be undertaken at this point? And what are the expected consequences of the action?
- The *future* of the trajectory deals with the projected course of action. This will be determined in part by the vision for the interaction, the plan or plans for how to achieve that vision, and who is involved in the planning activities

What this means for groupware. Groupware must allow people to manage and stay aware of their evolving interactions as they unfold over time. This includes a group's control over past, present and future aspects of routine and non-routine work. In terms of understanding the past, groupware must help individuals understand what has happened up until the current moment. This is especially critical for individuals who are latecomers to a groupware session. They have not had the privilege of witnessing all the past interactions. In addition, this information allows individuals to leverage past experiences. For instance, the use of change bars helps represent changes made to a document. These bars can be used to provide further information, such as the nature of the changes, the identity of the collaborator making the changes, and so forth.

Understanding the present means that groupware must provide awareness of the current actions that are taking place. Amongst other things, staying aware of others and their actions helps people coordinate and negotiate plans/activities and notice and repair breakdowns. In addition, this information aids with anticipating and predicting future events (especially routine work). This is strongly tied to the principles of mutuality outlined in heuristic 2.

HEURISTIC 5: Provide a way to organize and relate locales to one another

Theory. Within the Locales Framework, the *civic structure* considers each locale's relationships, interactions, and influences within the global context (i.e. beyond a person's known social worlds and locales). In dealing with civic structure, we must address:

- how the locale of interest is formed and populated
- how multiple locales fit together
- how to structure connections to facilitate encounters within and between locales
- how to facilitate the termination of social worlds

Forming and populating a locale. Both social worlds and locales are not created out of nothing. In some instances, locales are created from templates containing the standard resources to assist its members in accomplishing their task. For example, a template containing an agenda, a list of participants, discussion documents and so on can be used to create a new meeting locale. In other cases the characteristics of a

locale may be determined by its manner of creation. A group may appoint a sub-committee for some purpose, which would logically require its own locale. The 'sub-locale' would have a subset of members from the parent locale. It may also inherit some key documents from its parent and be automatically 'adjoined' with its parent and perhaps other related organizations. New social worlds can also emerge from arenas. Arenas arise when different social worlds interact around some issue (e.g. a contentious issue). New social worlds can arise out of these interactions as contentions are resolved and the worlds move closer together.

Fitting multiple locales together. Rather than existing as an independent entity, social worlds and locales usually have dynamic relationships with other social worlds and locales. Understanding how multiple locales fit together allows us to identify potential sources of external influences and inter-dependencies. Influences can be organizational, professional, financial, contractual, legislative, ethical, cultural, political, technological, and so on. Most social worlds, through their locales, create different interfaces for different civic structures relationships. This enables them to differentiate between that which is internal and private, and that which can be made available to everybody and anybody.

Structuring connections between locales. Within the civic structure context there is the issue of how we find and interact with people and locales beyond our own. For example, clinicians need the opportunities to interact with each other and build up peer networks, thus increasing trust levels. The way in which civic structures are established can facilitate discovery of other locales, people, and resources that may be of interest. The opportunity for discovery may be focused or through a chance encounter.

Terminating social worlds. As social worlds dissolve so will their associated locales. However, the results of a social world's work are often required beyond the life of that world. Maintaining residual objects after a social world has completed its collective task ensures the results of this work can be accessed at a later time or incorporated into other locales where needed.

What this means for groupware. Since locales are not typically created out of nothing, groupware must provide assistance with their creation. This may involve providing the necessary tools and applications to develop a new locale (e.g. through a template) or the ability to transfer information from an existing locale to a new one. An example is the way MSN helps people create their 'own' Web community. MSN provides membership policies (public vs. private), directory listing options (listed vs. unlisted) and a choice of categories and subcategories for the site. Listing your web community ensures that the site will appear in the community directory and search engines. This allows individuals with similar interests to find the site.

With respect to removing old locales, the groupware application must allow individuals the ability to save and reference the results of the social world's work at a future time.

Locales are rarely independent of one another. Consequently, groupware must provide a way to organize and relate locales to one another (civic structures). People also need a way to find their way between locales. For instance, social worlds may come in contact with one another through group collaboration or conflict. Under these circumstances, groupware must provide the means to communicate between locales. Or, social worlds may attempt to seek out one another. Searching mechanisms such as a computer-based search engines can help find the desired social world. When two social worlds come in contact, it may be convenient for groupware to employ a shared place that exists 'between' the two groups. Objects that need to be accessed by the members of each group can be put in this place.

Locales Framework Heuristics

Provide centers (locales)

Provide virtual locales as the site, means and resources for a group to pursue team and task work. The system should collect people, artifacts and resources in relation to the central purpose of the social world.

Provide awareness (mutuality) within locales

People and artifacts must make presence information available to others. People must be aware of others and their activities as they evolve.

Allow individual views

Individuals should be able to adapt their own idiosyncratic view of the locale or aggregate multiple locales in a way that reflect their responsibilities, activities, and interests.

Allow people to manage and stay aware of their evolving interactions over time.

People must be able to manage and stay aware of their evolving interactions. This involves control over past, present and future aspects of routine and non-routine work.

Provide a way to organize and relate locales to one another (civic structures).

Locales are rarely independent of one another: people need a way to structure the locales in a meaningful way, to find their way between locales, to create new locales, and to remove old ones.

A.7 Problem reports

O Provide the means for intentional and appropriate verbal communication O Provide the means for intentional and appropriate gestural communication O Provide consequential communication of an individual's embodiment O Provide consequential communication of shared artifacts O Provide protection O Allow people to coordinate their actions O Manage of tightly and loosely coupled collaboration O Facilitate finding collaborators and establishing contact	O GroupDraw O Groove
Description of usability problem:	
Severity rating: O Major obstacle to effective collaboration O Minor obstacle; people can work around it	
Solution (optional):	

O Provide centers (locales) O Provide awareness (mutuality) within locales O Allow individual views O Allow people to manage and stay aware of their evolving interactions over time O Provide a way to organize and relate locales to one another (civic structures)	O GroupDraw O Groove
Description of usability problem:	
Severity rating: O Major obstacle to effective collaboration O Minor obstacle; people can work around it	
Solution (optional):	

Appendix B: Heuristic Evaluation Usability Problems

This appendix provides the raw data used for conducting the analysis in Chapter 5. The raw data is broken down according to each groupware system as well as the two sets of groupware heuristics. It is presented in the following four tables:

B.1 Mechanics of Collaboration heuristics – GroupDraw

B.2 Mechanics of Collaboration heuristics – Groove

B.3 Locales Framework heuristics – GroupDraw

B.4 Locales Framework heuristics – Groove

Within each table, all of the inspectors' original problem reports (column 3) are grouped according to a known teamwork usability problem (column 2) derived through the results synthesis. Note, if a portion of any give problem report is in *italics*, this means that this portion of the report has been categorized according to the corresponding usability problem and heuristic. The rest of the report has been categorized according to a different problem (and possibly a different heuristic). The final column presents a unique identifier for the author of the original problem report.

B.1 Mechanics of Collaboration heuristics – GroupDraw

Table B.1 GroupDraw Problems reported using Mechanics of Collaboration heuristics

Heuristic 1 – Provide the means for intentional and appropriate verbal communications			
1	No means for verbal communication between individuals. This makes running commentaries extremely difficult. (M)	Can't communicate verbally except with phones.	2
		There is no facility for spoken verbal communication. It is possible to send notes and also simply use the text tool to write messages to each other, but it would be better to incorporate a module, which just allows spoken communication.	5
		No verbal communication.	7
		No facility for verbal communication. To do text communication, must click on Collaboration & Show Participants - not intuitive.	8
		If not phone, no means that are easy and natural to communicate verbally.	9
		No way to talk to other person (verbally).	10
		No direct discussions available (other than chat), no verbal communication. No verbal commentary. (Provide a digital audio link for voice chat)	12
		No support for verbal communication. Assumption appears to be that users are in close proximity to each other, or that they have themselves supplied a voice channel, like a telephone.	15
		No voice channel associated with the program	17
		No verbal communication supported. (Provide an audio channel)	20
2	Need to hit OK to close each new message that is received. This becomes cumbersome if you receive multiple messages. (m)	Inconvenient chat facility. In order to chat, one must send a note. This note appears as a small window on the other person's screen. What ends up happening is that too many windows appear on the screen. I found this particularly annoying because I had to click 'OK' a lot. (Chat facility should just have one main screen where all conversation is recorded.)	3
3	Difficult to keep track of the order of incoming messages especially when you are receiving multiple messages. (m)	Sending a message to another person produces a stand-alone pop-up message on the recipient's screen. If many messages are sent, the screen will become littered with these pop-up messages, and *it is very difficult to establish the time order in which they are sent.* (Display the messages in a single list).	4
		The only way I knew what order my messages came in was to look at the time stamp. I did not know what each message was referring to.	10
		Text message windows are cumbersome to view when multiple messages arrive. *Difficult to save history of messages when each is an individual window.* If window is maximized, new messages don't appear (only in taskbar). It is hard to catch them appearing.	12
4	Cumbersome to view and manage multiple messages. (m)	Sending a message to another person produces a stand-alone pop-up message on the recipient's screen. *If many messages are sent, the screen will become littered with these pop-up messages,* and it is very difficult to establish the time order in which they are sent. (Display the messages in a single list).	4
		Text message windows are cumbersome to view when multiple messages arrive. Difficult to save history of messages when each is an individual window. If window is maximized, new messages don't appear (only in taskbar). It is hard to catch them appearing.	12
5	Cannot reply directly from an incoming message. The user is forced to open a new message in order to reply. (m)	I could not simply reply to a message, I had to physically select the user and message them separately. (Have a reply button on the message pop up).	10

6	Easy to miss an incoming text message since they do not appear in a consistent place and may appear behind the main workspace. (m)	When seeing a note, it's easy for its window to get buried behind other windows and lost in the taskbar. (Make the representation in the taskbar flash to indicate a new message has arrived (though it shouldn't be too obtrusive), since there may be multiple messages blinking at any one time.)	1
		Chat facility was bad for initiating communication. Even though little notes popped up, it could never get my attention. If I was intent on drawing on one part of the screen, I wouldn't note anything else. My partner got so frustrated; she resorted to saying my name. (Provide some sort of sound every time a note appears)	3
		If User A sends a message to User B, and User B's screen is maximized, the message appears behind the screen and User B may not see it.	4
		My partner messaged me and asked if I could see that he selected something, I did not see the message until late and I missed the action.	10
		My partner messaged me (via chat) while I was writing this, I was unaware of the message until I looked up. (Provide a sound on incoming message).	10
		Text message windows are cumbersome to view when multiple messages arrive. Difficult to save history of messages when each is an individual window. *If window is maximized, new messages don't appear (only in taskbar).* It is hard to catch them appearing.	12
		Chat interface way too awkward. *Sometimes can't tell someone chatted to me.* (Always visible, audio cues, conversational awareness etc.)	18
		It seems to be random whether or not someone actually sees a message that I send out. (Have a message component on the GroupDraw interface somewhere.)	21
		Easy to miss a note - it pops up and is hidden by drawing on the note.	21
7	Impossible to simultaneously broadcast a message to multiple participants. (m)	The message sending facility allows a message to be sent only to one other person. This is a very restrictive limitation, as will not allow group conferences to take place. (Allow a single message to be sent to many users.)	4
		The "send note" facility only seems to be useful for one person to another. There is no way to tell everyone online the same thing, using notes at least.	5
		Can't send notes to multiple people.	9
		I could only message one person at a time i.e. I could not send the same message to more than one person without retyping it. (Have a chat window, or allow me to select message recipients in the message window)	10
		Impossible to broadcast a message to all.	22
		It is not clear whether you can send messages to the group as a whole - instead of just to specific users.	23
8	The window on which you type a note is a different screen than the note window. (m)	Chat facility - window on which you type a note is a different screen than the note window. Irritating because sometimes I want to read the conversation flow. (Collapse the 2 windows into one screen.)	3
9	Too many interaction steps in order to send a note to another participant. (m)	In order to send a note to someone, too much window navigation is involved: Select "collaboration" menu, then click on "picture <user>" and choose "Send note", then type in note and push send. (Have a list of participants in the GroupDraw window, double-click someone's name to send a note to them.)	1
		Inconvenient chat - one must use the mouse and click a button to send the message. Just annoying once your hand has been positioned to type on the keyboard. (Option of hot key to send message.)	3
		The "send note" functionality is very clunky. It isn't useful for intentional, quick response communication.	5
		Text message is difficult to find and use. Not clear to right-click a button. *After selecting message box it is not active, so user must click in it.*	11
		Means for verbal communication does exist but doesn't seem intuitive and *takes a while to do it*. (Provide a chat area so user doesn't have to go far to talk with others.)	13
		Chat interface way too awkward. Sometimes can't tell someone chatted to me. (Always visible, audio cues, conversational awareness etc.)	18
		Difficult to send a text message through participant list - 4 to 5 actions to send message.	22

10	Awkward sending notes since text communication happens in windows outside of the workspace. (m)	Inconvenient chat facility - The chat facility was on a different window from the GroupDraw. Thus, it was too annoying to actually use it. So, we ended up communicating via text labels on the actual drawing itself. When finished we deleted the text labels. (The chat facility could be provided on the GroupDraw window.)	3
		We find that it was awkward to switch context to the "messenger" box to send messages to each other. We would rather type it into the GroupDraw window itself, as it felt more convenient. However, clutter began to build and we start erasing. Only fear is that we would accidentally erase things unintentionally.	6
		Chat utility is separated from drawing area - must manage separate windows.	19
		That to message someone involves using another window is annoying. (Merge interfaces. Chat toolbar.)	21
11	Controls for messaging are difficult to find. (m)	No facility for verbal communication. *To do text comms, must click on Collaboration & Show Participants - not intuitive.*	8
		Hard to find where to send a note - have to search for it.	9
		Text message is difficult to find and use. *Not clear to right-click a button.* After selecting message box it is not active, user must click in it.	11
		Means for verbal communication does exist but *doesn't seem intuitive* and takes a while to do it. (Provide a chat area so user doesn't have to go far to talk with others.)	13
		The 'chat' feature is not immediately obvious - i.e. it is not clear how to send messages.	23
12	Others are not aware of side conversations. (m)	When a message is sent to another user, the remaining users are not aware of that conversation is taking place. This may contribute to a sense of isolation, as a user may think little collaboration is taking place, when in fact there are many user-to-user conversations. (Place an animated "talk bubble" beside a user's icon whenever the user is taking part in a conversation.)	4
		Third party can't overhear conversation to gather information about what others are doing or going to do. (Use digital audio link to broadcast voice conversation to other users or use chat room with visibility for all users)	12
13	Difficult to establish communication for conferences involving >2 participants. (m)	The lack of voice is especially problematic when there are >2 people as conference calls by phone are difficult to set-up for most people.	17

Heuristic 2 – Provide the means for intentional and appropriate gestural communication			
14	Emblems and illustration are not possible. (M)	*Lacks the means for intentional gestural communication of illustration.* Bad because there is no way in which I can indicate how big I want something to be drawn. I can't just go to the chat facility and say that I want the box to be 25mm x 25mm. That makes no sense. Big problem cuz this is a drawing application.	3
		There doesn't seem to be any means of expressing illustration. We only have one mouse pointer for ourselves, so this introduces a limitation.	5
		Illustration and emblems have to be communicated outside of the system.	7
		Emblems are not possible unless another channel is provided (e.g. video). Even if it were, the emblem can be easily missed if the person is looking at the workspace rather than the video.	17
15	Cannot indicate how big something should be drawn. (m)	Lacks the means for intentional gestural communication of illustration. *Bad because there is no way in which I can indicate how big I want something to be drawn. I can't just go to the chat facility and say that I want the box to be 25mm x 25mm.* That makes no sense. Big problem cuz this is a drawing application.	3
16	Deixis is extremely difficult with the text chat facility. (M)	Difficult to gesture where I want to put something. Must indicate thru chat for other person to watch my cursor. Even then, other didn't understand & we had to agree that if I stopped moving for more than 1s that's where to put it (deixis).	3
		If both are moving a line and want to drop it, no way of intentional gesture to indicate unless you leave space to chat.	3
		Due to the lack or a module for verbal communication, it becomes apparent that it will be very difficult to indicate deixis in this system. How can you say "this one is too big" if you can's say anything.	5
		I wanted to point to a picture and say something but I could not do both at the same time.	10
		Verbal and gestural communication is uncoupled. Limited support for intentional gestural communication through motion. (Users can create their own language of gestures.)	16

16	Cont'd	*Deixis is possible, but disjointed - have to text chat a message "Over here" and then move telepointer.* May or may not notice. (Message that can attach itself to telepointer or voice chat.)	21
17	May not notice textual deixis references since it is displayed outside of the drawing context. (m)	Deixis is possible, but disjointed - have to text chat a message "Over here" and then move telepointer. *May or may not notice.* (Message that can attach itself to telepointer or voice chat.)	21
18	Cannot point to the menu. (m)	Other user's cursor doesn't appear over a menu in another person's window. (While it probably won't be desirable to show the actual menu expanding in another person's window, you may want to show what menu the other person is looking through)	1
		T*elepointers used for gestures are limited to only the drawing board, not the menu bar* or icons. (Allow telepointer to be visible throughout entire application)	12
		Telepointers do not point outside workspace, so difficult to point to tool buttons, *menu items.* Telepointers stick to the point where you left the workspace, which can be misleading.	22
19	Cannot point to the tool palette. (m)	You can't tell if others are going to click on a drawing tool - telepointers don't leave the white area. (Let the telepointer leave the white area)	8
		Telepointers used for gestures are limited to only the drawing board, not the menu bar or *icons*. (Allow telepointer to be visible throughout entire application)	12
		Telepointers do not point outside workspace, so difficult to point to tool buttons, menu items. Telepointers stick to the point where you left the workspace, which can be misleading.	22
20	Cursor's shape cannot be changed to reflect the intended meaning behind a gesture. (m)	No gestures available via the cursor. Other person appears as a dot, and will never change. (Allow another user's cursor to change shape on your screen (e.g. arrow) to allow gesturing)	1
		No way of grabbing partner's attention except through the chat facility.	3
		The telepointer of another person always appears the same, regardless of whether he is pointing something out, drawing, moving an object, or deleting. *This can make it somewhat difficult to figure out exactly what he is doing.*	4
		You only have one type of telepointer, so can't change it to say "this one". (Allow users to change shape of telepointer to point things out).	8
		Hard to read gestures from a telepointer. The telepointer stays the same shape even if someone is pointing to something or gesturing, don't know this what they are doing. No deixis between people and objects as a result. (Create different telepointers to allow for "obvious" pointing or different gestures, this will give deixis. Need to also link voice to the gestures, referring to earlier problem.)	12
		Telepointer is not an arrow shape, so may be difficult to tell which way it is pointing.	22
21	Difficult to make a gesture that attracts another's attention. (m)	No way to bring users attention to area by highlighting	9
		No indication can be made towards a specific item a user is working on. That is, if there are many users on the system, one specific user cannot direct others' attention to his/her work. (Change mouse pointer so that it creates a pointer.)	14

Heuristic 3 – Provide the consequential communication of an individual's embodiment			
22	Actions can go unnoticed since the telepointers are small. (m)	Telepointer too small for partner to sometimes see.	3
		While telepointers are present they are small.	17
23	Tough to identify the telepointer's owner from its appearance since they all look the same. (m)	With >2 people using the system, don't know which cursor belongs to which person.	1
		Can't tell who is who - no reference between multiple participants and their telepointer. (Colour match list of participants to telepointer colour.)	2
		Not always unique identifier. GroupDraw allows for same colour pointers.	3

23	Cont'd	Cursors are all the same colour, size, and shape, so it is not possible to associate a cursor with the user who is controlling it.	4
		Telepointer - Don't know who is associated with a pointer easily. Colours can help, but one has to remember.	7
		Sometimes multiple users' telepointers are the same colour - no way to tell the users apart. It appears the last person to join sees different coloured telepointers while the others see the same colour.	8
		Can't tell who she is - if more than 2 users will colours of telepointer be different? - Associate something with it.	9
		Unable to tell multiple users from one another, as all pointers look the same. (Make them a different colour for each user.)	11
		Telepointers appear the same for all users and do not distinguish between users. Unsure of who is attempting to communicate a gesture. (Create different telepointer for each user. Possible different colour or shape for easy distinction.)	12
		Can't distinguish which person is manipulating an artifact or what their identity is. Can't communicate with someone about what they are doing if you don't know who is doing it. (Create unique telepointers that uniquely identify a person)	12
		No differentiation between the cursor colours for each user. (Different colours or pointer styles for each user.)	14
		It is hard to identify who a cursor belongs to from its appearance. While colour can be changed, I suspect that people will just stick to the default, meaning cursors will look identical. If colour is changed, I suspect that it will be difficult to associate a cursor with a person strictly by recognizing/remembering this pair especially if there are more than 2 people.	17
		Cannot tell which telepointer belongs to whom. (Differentiate telepointers visibly by user.)	18
		Telepointers do not indicate the user's name, so are difficult to coordinate with. (Username beside each telepointer.)	19
		Can use telepointer to gesture but it is hard to tell one pointer from another.	20
		Telepointers provide embodiment but they aren't distinguished enough - everyone else's looks the same. (Provide a unique telepointers.)	20
		While you can see an object manipulated, you can't tell who is doing what with it - unless you are warned with a message.	21
24	Cannot tell via the telepointer where other participants are looking. (m)	*Don't know other's view and gaze.* My window was smaller and every time partner drew off my screen, seems like he disappeared. (Provide an overview of entire workspace).	3
		No gaze awareness to see what others may be thinking about doing or about to do. No means to establish eye contact. Actions are not coupled to the workspace. (Just wait to see once a person starts working, or ask in chat message. Video to see gaze awareness)	12
		The telepointer is a weak embodiment. We cannot tell where the other(s) are looking, although a moving telepointer probably suggests that the other is looking at it.	17
25	Cannot establish eye contact with other participants. (m)	No gaze awareness to see what others may be thinking about doing or about to do. *No means to establish eye contact.* Actions are not coupled to the workspace. (Just wait to see once a person starts working, or ask in chat message. Video to see gaze awareness.)	12
26	Cannot determine an individual's mode via the telepointer (e.g. cursor does not show that a participant has selected the square button and is about to create a square). (m)	Doesn't show what tool (line, circle, rectangle, text) the other person has selected. (Change the other person's cursor from a dot to a dot along with their tool)	1
		The telepointer of another person always appears the same, regardless of whether he is pointing something out, drawing, moving an object, or deleting. This can make it somewhat difficult to figure out exactly what he is doing.	4
		Cursors of other users are not visible when they grab for a tool in the tool menu. *When the cursor reappears, I have no idea what tool the user selected.* (Allow other users' cursors to be visible everywhere all the time. Also allow cursors to change shape to reflect the tool selection.)	4
		Cannot determine what tool is picked by other participant. Can only realize that when things are drawn on screen. (Show what tool is picked)	6

26	Cont'd	There is little information about what the other user will do next, aside from the telepointer's position. Thus little unintentional information can be given. (Change shape of telepointer depending on what tool the other user has selected.)	8
		I was unaware of what the other person was attempting to do until they finished the action (i.e. drawing line, circle, message)	10
		User's cursor remains the same at all times, making it impossible guess/infer what action they are going to perform until it is done or in progress. (Change the telepointer on all displays to show that tool each user has selected.)	11
		While intentional gestural communication is supported, consequential is not (or at least, is supported poorly). I can see other's cursors, but they are context free - I cannot tell from looking at a cursor what action that person is likely to perform next. (Change the pointer icons so they reflect which tool a person is currently using.)	15
		Palette selections are invisible to others, thus we *do not know what 'tool' the person is holding* i.e. what state they are in.	17
		No way of telling what someone's going to draw. (Telepointers changes with tool.)	21
		No indication of what tool has been chosen. (Mode-specific telepointer symbols.)	22
27	A still telepointer is problematic, as we don't even know if the person is still present, whether they are attending to the display, in text mode, etc. (M)	When a user's cursor moves off the GroupDraw application, its telepointer will still appear on the GroupDraw canvas in another user's window. Thus one person may believe that another user is engaged in GroupDraw when they are actually using another application, or away from the computer itself. (Provide some means to indicate that a user is away from the program for a period of time.)	1
		Sometimes, not always evident as to where a person is located. If typing into chat facility, no way for me to know that cuz cursor is not on the screen. Don't know if person has left.	3
		If somebody does make their screen bigger and goes off the edge of your smaller screen, their telepointer remains at the last valid location on your screen, not moving off at all.	5
		When the person goes for a control/choose something from the side - *can't tell if they have stopped work* or what they are choosing.	9
		Can't tell when the user is not working any more in the screen unless they tell you.	9
		When other person's telepointer is idle, it is not easy to determine what he's doing without sending him a note Did he leave? Is he writing a note?	13
		There is no embodiment other than the position of the remote cursor. As long as the software is running you don't know if the person is still there when their pointer is not moving.	16
		A still telepointer is problematic, as we don't even know if the person is still present, whether they are attending the display, etc.	17
		While you can tell if someone's in the workspaces, there is no way to tell if they're actually there if they are not moving. e.g. could be reading e-mail. (Change telepointer colour based on user's active window.)	21
		Telepointers do not point outside workspace, so difficult to point to tool buttons, menu items. *Telepointers stick to the point where you left the workspace, which can be misleading.*	22
28	Telepointer will remain in the spot where the participant has left the shared space. But if the participant were to re-enter the shared space at another location then the cursor jumps to that new location. (m)	It is possible for a user to move his cursor off the left (or top) of the workspace and move in again from the right (or bottom). On other people's screens, this causes the user's telepointer to jerk suddenly from the left (or top) to the right (or bottom).	4
29	The text cursor does not appear on the remote site when people are typing or editing an existing text object. (M)	When text is selected and mouse pointer is moved away, can't tell that text is about to appear. (Provide a text cursor)	2
		Can't tell that the other person is going to start typing - no coordination, doesn't communicate intention, and the appropriate gestural communication isn't there.	9
		Text cursor does not appear on remote site, thus people don't know that the other is in 'type' mode.	17

29	Cont'd	As text is being typed, the remote text cursor does not appear. Thus others cannot see a person moving through the text with the arrow key.	17
		Caret is not shown on other screens when using the text tool.	22
30	The remote person cannot see when a person has finished editing the text object (i.e. by clicking outside). (m)	Similarly, the remote person cannot see when a person has finished editing i.e. by clicking outside.	17
31	Cumbersome if you want to draw the same object multiple times since you have to reselect the tool every time you have finished drawing the item. (i.e. toolbar button does not remain in the locked position) (m)	In GroupDraw, tools have to be reselected every time, as they don't stick. So this causes one to click on the tool every time even though the same tool was just used. It interferes with smooth operation of collaboration.	6
		No way to keep a tool active for more than one action. Prevents or slows collaboration. (Should be able to keep a tool selected.)	21
32	The telepointer remains on the screen when typing or editing text; however, it is not located in the same spot where you are typing. (m)	I typed text in a completely different location than my mouse pointer was. This was confusing to other person.	2
		Writing text in a place other than the mouse pointer makes you think other person not currently typing. Doesn't allow you to coordinate actions.	9
33	Text chat does not support cues coupled to the conversation (e.g. tone) to be picked up by the receiver. (m)	Chat feature is limiting, can't hear tone of expression in voice. Actions are not coupled to conversation. (Use digital audio link for voice chat along with text messaging or use symbols to annotate expression)	12
		Chatting via the 'Send Note' mechanism doesn't always allow voice intonation and the 'mood' behind a message e.g. are they being sarcastic or serious?	13
		Only text can be used to communicate ideas. Poses a problem because written words are not always interpreted the same as language. (Use telepointers or build on microphone transmission to communicate ideas)	14

Heuristic 4 – Provide consequential communication of shared artifacts			
34	When selecting an object, handles appear locally around the object; however, they do not appear remotely. (M)	Can't see the handle bars of an artifact when others move or resize it.	2
		Embodiment - can see other but don't know what they are doing. E.g. see mouse move back and forth but that's all. In reality, other person attempting to move rectangle. (When attempting to move/pick up something, perhaps telepointer can turn into a hand or something with a similar idea. Thus this indicates the other person's intentions and actions.)	3
		While the telepointer shows where the other person is, there is nothing to indicate that the other person has selected an item in that area.	5
		One user cannot tell when another user has selected an object or set of objects. This is the only obvious feedthrough lacking in this system.	5
		There is no indication of one participant selecting an object until he/she moves it. If it so happens that multiple participants grab the same object but moving it in different directions, the pointers get a little messy. (Give an indication that an object is selected)	6
		No indication that someone else has selected something or intends to manipulate it. When someone does manipulate an object, then others can see. (Change the telepointer to show action. Show selection)	7
		Can't see that a person's grabbed an object only when they manipulate it can you tell or how they're going to manipulate it.	9
		If I selected an item on the screen, I noticed that I selected it, when my partner selected something the only way I knew was when the object moved. (Show selection boxes in the colour of my partner's pointer)	10

34	Cont'd	It is unclear when/if an artifact is being manipulated by another user until it begins moving. (Highlight object when it is selected to show it is active by another user.)	11
		Clicking on an object - while handles appear locally they do not appear remotely, thus the others do not know that the object was selected.	17
		While you can see when someone moves something, you can't see when they select it. When two people move something at the same time, the results seem unpredictable. (Show remote handles, lock objects)	21
		No indication that an object has been selected by another person. (Show selection handles to others.)	22
35	The telepointer appears 'distant' from the object being manipulated on the remote screen since handles may not appear right on top of the selected object (i.e. a circle). (m)	When creating or moving an object such as a circle, the handles are not shown on the remote machine when moving and not on both for creating. Thus the telepointer is 'distant' from the object being moved.	17
		When using circles, telepointer is not touching the circle - could be misleading. (Show handles.)	22
36	From remote a person's perspective, it is tough to identify the individual performing the symbolic action. (m)	The delete action, from remote people's perspective, is completely disassociated with the deleter as the handles are not visible.	17
		No way to tell what commands others are doing unless they use their telepointers.	24
37	Symbolic actions occur extremely rapidly and can be easily missed. (M)	Allows for users to manipulate and move objects but removal is hard to figure out. Not to mention the fact that users can delete one another's objects. (Have a switch which allow users to set whether they allow others to manipulate/delete their objects)	14
		Deletes are extremely rapid and can easily be missed.	17
		Most types of actions provided with good feedthrough except deletions - object 'winks' out too quickly. (Try employing an exaggerated action e.g. super-nova)	20
38	There are no sounds accompanying the manipulation of items within the workspace. (m)	Can't hear that an artifact is being manipulated. It may not be in your direct view. (Add sound when manipulating objects. Just use visual or people manipulating)	12
39	Local palette selections are invisible to others. (m)	When I click on a toolbar item, others don't see it - my intentions are hidden.	2
		When I double-click on an icon to lock the type of tool, this is not indicated to others.	2
		No indication that others clicked on menu items.	2
		Telepointers only stay within white area. Don't know exactly what other person selected (line, circle, etc) when he goes to select a shape.	3
		Cursors of other users are not visible when they grab for a tool in the tool menu. When the cursor reappears, I have no idea what tool the user selected. (Allow other users' cursors to be visible everywhere all the time. Also allow cursors to change shape to reflect the tool selection.)	4
		Telepointer is only in white area - don't really know which tool a person is about to use because the tools are very close together and the pointer moves fast. There is no anticipation of what people are going to do (Show in the telepointer what tool people are using. Share the tool box and menu as well)	7
		You can't tell which tool the other user has selected. (Highlight the tool the other person has selected)	8
		When the person goes for a control/choose something from the side - can't tell if they have stopped work or *what they are choosing*.	9
		Can't see the selection/blackening of palette selection from other user (Different colour for select)	9
		Cannot see other users clicking on toolbar items. Slows down communication and other users are unsure of a specific user's intended actions. (Show mouse hovering over toolbar)	14

39	Cont'd	Related to the previous point, I can't see when other users access the menu. I can see them moving towards the menu but they might just be headed for that end of the screen. (Being able to see others' cursor over my menu would help, even though their actions would have no effect on my menu. I would at least know for sure where they were.)	15
		Palette selections are invisible to others, thus we do not know what 'tool' the person is holding i.e. what state they are in.	17
		Similarly, menus are invisible thus we do not know others intentions to (say) leave.	17
40	Difficult to determine who created and/or last edited an object since all Items are the same colour regardless of who created them or is working on them. (M)	Difficult to remember who created and/or last edited an object once the canvas gets cluttered with shapes.	1
		Artifacts are the same colour, texture regardless of who created it or is working with it.	2
		Can't tell who changed something.	7
		When someone enters a meeting there is no way of determining who was working on what. (Provide different colours for each user's drawing)	14
		No way to signal value or importance of items.	18
		Objects do not convey sense of ownership. (Somehow visually mark the ownership/authorship of objects.)	18
		Objects don't say if/how/when/ by whom of changes to them.	18
41	Artifacts do not convey a sense of history. (m)	When many users work together on the same form, it is currently difficult to tell how often a particular field has been edited (you may have originally written the text into the field, so you are interested in knowing whether many changes have been made).	4
		Share artifact - Can't tell that something has been deleted or moved since the last time you looked on the screen.	7
		Can't tell what has been done to a shape when you have stepped away or worked on something else. (History)	9

Heuristic 5 - Provide protection			
42	Two individuals can simultaneously move and re-size the same object, which causes erratic behaviour by the system. (M)	If 2 or more people attempt to alter the same shape at the same time, confusion will result from the rapid flickering that occurs when the people are dragging the object at the same time. (Lock the handle of the shape that is being used so that other's can't grab it, and make this condition visible to all.)	1
		Both can drag same item at same time. Item bounces back and forth between people.	3
		There is some erratic behaviour when two users choose points on the same line and one goes up and the other down. There is a lot of flickering and an odd outcome.	5
		There is no indication of one participant selecting an object until he/she moves it. *If it so happens that multiple participants grab the same object but moving it in different directions, the pointers get a little messy*. (Give an indication that an object is selected)	6
		If someone draws something and then other person moves it - interferes with their work. (Have a way for someone to visually define work area so that others see they are working on it alone)	9
		When my partner and I tried to move something together, I could not figure out what affect I had on the object and what affect he had.	10
		Objects drawn on the canvas can be edited/manipulated by more than one person at the same time.	13
		In real life, if two people are moving the same object the two people move with the object. Here there is an elastic that places the object between the two users. (Make motions additive. Motion of one mouse drags the other with it.)	16
		As long as the user can grab hold of one of the objects being worked on, then they can be interfered with.	16
		While you can see when someone moves something, you can't see when they select it. *When two people move something at the same time, the results seem unpredictable.* (Show remote handles, lock objects)	21

43	Two people can edit a text item at the same time, which results with the items being inconsistent across the two systems. (M)	Two people can modify the same text box, but it loses one person's changes. (Either lock the text box when one person starts modifying it or show modifications in real-time)	5
		Shared text boxes are glitchy, when they are manipulated by more than one user. May not get feedthrough of others' changes and others' changes may not be applied. (Fix the glitch or set a rule of not sharing the editing of text.)	16
		Interference within the text tool is inconsistent. It is unknown who's changes will be applied.	16
		There is a bug where multiple entries by different people into the text object don't do the correct thing. i.e. some text disappears.	17
44	Collaborators are unable to 'correct' or 'undo' any conflicting actions. (m)	No undo operation.	1
		There is no undo functionality, so if another person deletes my object there is no way I can get it back. For the purposes of GroupDraw; however, all the objects are fairly simple and could be easily recreated (with the exception of a long, carefully written textbox message), so this problem is not very serious.	4
		There is no undo function.	5
		There is no undo feature; participants feel that they must not make any mistakes - pressure!	6
		Deleting other people's stuff - *how to recover* or protect.	9
		There is no undo function. Users can change or completely remove work done by themselves or others. (If object belongs to another user, notify them that it's being modified and allow them to decide if they'll accept changes.)	11
		No undo capabilities if someone deletes what you are working on. (Rely on social protocol that if they see you working on an object they won't touch it, but if you leave it for a minute they may delete it. Provide an undo capability. Access control can be placed on certain objects by a user so they can't be manipulated.)	12
45	One participant can delete an object while another is editing or moving it. (M)	Deletion of artifacts created by others is not challenged.	2
		Can delete something while other person dragging.	3
		As I am manipulating a drawing object, another user is free to delete it. (An audio channel would be useful here, so users can easily communicate their intentions. As well, the user who first draws/edits the object could be given sole ability for deletion until he lets go.)	4
		Possible to delete something that someone else is manipulating.	7
		Deleting other people's stuff - *how to* recover or *protect*.	9
		I was able to delete some artifacts that my partner had put on his screen. He had no idea what happened.	10
46	No indication that certain objects are not selectable. (m)	Can't select some shapes - doesn't warn you or give reason	9
		While people don't seem to be able to click the same handle on a line, no feedback is provided that one is 'locked out'.	17
47	No ability to protect personal objects to prevent others from modifying it. (m)	Can't stop a person from interfering with your own work - can't grab things back.	9
		No protection provided. No sense of ownership of objects. (Make security levels user configurable)	20
		There are no controls over who owns what objects. Every time something is manipulated, someone else can manipulate it if they catch it.	21
		There is no sense of ownership. While someone is typing a message, someone else is already moving the object.	21
		No protection of objects.	22
		There is no protection - difficult to say whether this is a real problem, more of a contextual issue.	23

48	It is difficult to implement social protocols as a means for protection given the chat facility and limited awareness. (M)	Social protocol to provide protection needs an audio channel for fast communication. Text chat is not effective enough for this. (Provide audio channel for voice chat in real time)	12
		Since it is hard to see intentions (largely due to lack of handles), we can inadvertently try to manipulate an object someone else has selected.	17

Heuristic 6 – Provide management of tightly- and loosely- coupled collaboration			
49	Tough to identify when tighter collaboration is necessary. (M)	Aren't aware if people need help without sending an explicit message. (Provide audio channel to ask for help. Overview of entire workspace with indication of what users are doing in their own space.)	12
50	If people have different size windows, the person with the smaller window cannot see actions & telepointers outside of their viewport. (M)	Different window sizes between users will cause some things to not appear at all on the smaller window, and that person will not know how that objects may be present, though not visible. (Provide some sort of overview of the entire drawing space.)	1
		Window size - when I reduce the size of my window, I cannot see the other person's embodiment.	2
		Window size - can't coordinate activities when I can't see the other telepointer.	2
		Don't know other's view and gaze. *My window was smaller and every time partner drew off my screen, seems like he disappeared.* (Provide an overview of entire workspace).	3
		Not always able to identify what others are doing. If in a different workspace, don't see them at all. Thus can't see what development they have made. (Provide an overview of entire workspace)	3
		No overview - If I work in a different place (my window shape is slightly larger), no way for other person to keep track of where I am and what is happening in my workspace. Thus poor management of tightly and loosely coupled collaboration. (Provide overview)	3
		Does not support collaboration well. Partner said he was going to draw a tree (his window larger). He drew a tree but in another workspace I couldn't see. No way for him to gesture where he was or where he was to draw the tree.	3
		If one user's workspace is much larger than other users' workspaces, cursors which move off the right or bottom of the smaller workspaces will mysteriously disappear in the larger workspace.	4
		One participant may resize the GroupDraw window and work on a new area, which will not be seen unless the other participant(s) also resizes the window in the same way. If so, then there is no privacy, and others may think that "rough" work is for everyone to see. (Include a personal window for each participant with an option to share when ready)	6
		No awareness of what others are doing outside of your own viewing area.	7
		You can't tell where the other people are if they move off the screen - you have to enlarge the window to see what they've done.	8
		Can't tell where other person is when screens different sizes. (Have an overview)	9
		I made my window big and drew something in the far edge. When I asked my partner what he thought he said "What picture?". His window was not big enough to see my drawing. (Show a context view of each participant's drawing/window)	10
		Unable to tell if the other person has left the window or is working in a different area. A person can enlarge their workspace without the other's knowledge (no way to tell). (User can either consider the space their own or assume the others can see it and expect they are following. Implement scroll bars that show up if a user increase their workspace.)	11
		Can't see others manipulating objects if their window is a different size than yours and they are working off the screen. (Provide a common view of the workspace, WYSIWIS. Provide a common overview of entire workspace)	12
		No overview to see all of possible workspace to see what others are doing all the time. (Provide overview of the entire space with local view)	12

50	Cont'd	People with different window sizes will have different views of what's going on e.g. someone with smaller window can't always see the other person's telepointer. (Provide an overview window that shows entire workspace to allow people to see if their space is different from others.)	13
		Cannot see users typing that is off of your current screen. (Provide a smaller image in the corner that shows work being done by other users)	14
		There are no transitions between tightly and loosely coupled collaboration. The entire space is tightly coupled unless your window is bigger. *In which you disappear from the other user.* (Provide large scrollable workspace with overview)	16
		If people have different sized windows, person with smaller window cannot see actions/pointer of others in unseen area. Therefore awareness of others is lost, inhibiting the move to tightly coupled work.	17
		If someone is working in a larger window, I can't see if they are working off my field of view. I get a message about something they did and don't understand. (Provide an overview.)	21
		If windows are different sizes, some actions/artifacts will be unseen by some people. (Visibility or indication of location.)	22
		Difficult to keep track of others actions if they are in a different part of the workspace - i.e. outside my view.	22
51	No indication that somebody has resized the workspace. (m)	Window size - no notice given to other participants that my environment changed.	2
		There is no indication when the other person has re-sized their screen (workspace). You may not have any idea that you are missing something important.	5
		We have no indication if both users are working on the same sized space, and if one has gone off the edge, or if they're just sitting on the border. (The border beyond which the user passed, could be lightly highlighted to indicate their workspace is larger.)	5
		On the same topic of resizing window, if a participant resizes his/her GroupDraw window, it does not show any of such indication to other participants. If the participant starts drawing in the new area, others will not see this.	6
		Cannot tell that others are working on a smaller window/different workspace.	7
		If windows different sizes can't tell when other user resized their window - can't tell that other user only moving in smaller area.	9
		Unable to tell if the other person has left the window or is working in a different area. *A person can enlarge their workspace without the other's knowledge (no way to tell).* (User can either consider the space their own or assume the others can see it and expect they are following. Implement scroll bars that show up if a user increase their workspace.)	11
52	Difficult to occupy a private part of the workspace in order to perform private work. (m)	One participant may resize the GroupDraw window and work on a new area which will not be seen unless the other participant(s) also resizes the window in the same way. *If so, then there is no privacy, and others may think that "rough" work is for everyone to see.* (Include a personal window for each participant with an option to share when ready)	6
		Can only have own workspace in the top left corner of screen (done by resizing the window). One cannot focus on another part of the screen.	7
		You can draw over work the other user has done, obscuring it.	8
		User can resize window and move to different portion to do private work, but if another's window is the same size they may not be aware that the work is private. (Allow alternate views of workspace i.e. public and private. Facilitate drag and drop between spaces. Section borders for private work, can move border - acts like a fence)	12
		Although people can work on different parts of the GroupDraw sketchpad, it is a small space, and there is no way to explicitly mark off an area and say "this is my space, I will join you in a moment."	15
		There are no transitions between tightly and loosely coupled collaboration. *The entire space is tightly coupled unless your window is bigger.* In which you disappear from the other user. (Provide large scrollable workspace with overview)	16
		The window has no scrolling, thus it is hard to occupy an 'individual' space unless the window is large.	17
		No capability to carry out individual work, which may be desirable in some cases.	23
		Cannot work privately in public space.	24
		Cannot work privately - all work is shared instantly.	25

Heuristic 7 - Allow people the ability to coordinate their actions			
53	Explicit fine-grained coordination is extremely difficult with the chat facility as the means of communication. (M)	The lack of spoken communication makes coordination of activities a bit more difficult, but not entirely impossible.	5
		Initial coordination is hard. Coming up with something to draw is difficult because no verbal communication. But one can keep out of each other's way and share work very easily.	7
		With a difficult to use text chat feature and very little communication through the telepointer. There is almost no way to coordinate actions between users.	8
		No easy voice communication to coordinate activities. Chat/messaging is not comparable or useful enough. (Provide audio channel)	12
		The other common means for these types of systems - text chat - is inappropriate for fine details of communication e.g. shadowing is hard and you can't type while drawing.	17
		Coordination is difficult, for many reasons previously stated. Most important are: *1) no audio for fine grained coordination* and 2) difficult to anticipate what others are going to do because handles and states are invisible and some actions (e.g. delete) are too rapid.	17
		Difficult to coordinate actions.	23
		No effective means of coordinating actions.	24
54	Hard to anticipate what others are going to do because remote handles are non-existent, states are invisible, and some actions (e.g. delete) are too rapid. (m)	We cannot see a person's intentions to create/move an object - due to not seeing handles and to not knowing what tool state they are in.	17
		Coordination is difficult, for many reasons previously stated. Most important are: 1) no audio for fine grained coordination and *2) difficult to anticipate what others are going to do because handles and states are invisible and some actions (e.g. delete) are too rapid.*	17
55	Difficult to communicate and collaborate concurrently since text and workspace functions are not tightly integrated. (M)	it is very difficult to coordinate actions with the built-in "instant messaging" facility. It is not possible to type my intentions or request for immediate assistance at the same time that I am drawing or resizing my object. (Provide an audio channel, allowing real-time comms)	4
		Hard to write and do things at the same time.	9
		Every time I wanted to message my partner, I had to stop drawing and move my pointer off the drawing surface. (Allow voice, or have the message/chat window pop up with a hot key or any time there is keyboard activity)	10
		I had to stop drawing when I got a message.	10
		Interaction is modal. Users must communicate their intentions through text, then perform the action. Does not allow communication while action is being performed e.g. a user can perform one action at a time. (Use third party software or telephone to verbally coordinate efforts)	16
		Verbal and gestural communication is uncoupled, making it difficult to coordinate actions. (Simultaneous voice communication)	16
		Very hard to coordinate actions. Tough to collaborate and do a drawing. No means or easy communication provided.	20
		When lots of activity is happening, it is impossible to quickly convey intent. (It is easy to merge chat and draw interfaces.)	21

Heuristic 8 - Facilitate finding collaborators and establishing contact			
56	Difficult to invite members to a new conference or a conference in progress; especially if they are not logged onto the system. (m)	I created a new conference and had to go to my old one to invite my partner to join.	10
		I could not invite someone that was not in the Open Registration i.e. cannot invite people who are not logged on. (Provide e-mail or phone feature)	10
		Can't message a colleague if they are not registered. What if you want to ask them to go on? (Send e-mail or phone them)	12
		How do you find contact information for people not currently running the program? Or what if they are registered but have left their desk for a while. You could send a note but you'd never get a reply or any indication that it hasn't been read.	13
		At the beginning there are different groups that people can join but there is no way to invite new users into the group drawing. (Allow a viewing of all users on the systems from which one can send a note to invite them to join the task.)	14
		"Finding collaborators and establishing contact" exists but it is clunky (Collaboration/Show Participants). The Show Participants label is a bad one; 'participants' implies people I am currently collaborating with, while 'users' (in the GroupDraw window) seems like it ought to be the list of all users in all spaces. In fact it is the other way around. (Rename the menu options, and enable right-clicking on names to send an invitation. That is easier than the hunt-through-the-menus approach GroupDraw currently employs)	15
		There is no means to find out how to establish a voice channel with the other person(s). e.g. phone numbers, groupware, etc.	17
		There is no 'easy' way to invite people in.	17
		Unless person is already in the conference, there is no way to communicate with them. (Need a rendez-vous point.)	18
		Facilitate finding collaborators and establishing contact - if user knows how to find others (which menu to click) this is OK. (Should provide this info automatically)	20
		There isn't a way to invite a person to a sketch session from a sketch session. (Collaboration - invite users - list from registrar.)	21
57	No way to determine somebody's state or availability for collaborating. (m)	Multiple parties have to be willing to participate - no facility to notify another participant or see if they are busy	2
		There is no indication for whether a user wishes to be contacted or left undisturbed. This makes it difficult for me to describe whether now or later would be the best time to communicate with the user. (the icons for the user could incorporate a "door state" idea)	4
		Cannot see the state of the other person. So we don't know whether he/she is going to do something, or the fact whether he/she is even on the computer. (Status indication - like messenger programs)	6
		Cannot be logged on and indicate that one is not available.	7
		How do you find contact information for people not currently running the program? *Or what if they are registered but have left their desk for a while. You could send a note but you'd never get a reply or any indication that it hasn't been read.*	13
		It is unknown if a user is available unless you send them a note, and they respond.	16
		We do not know if the other's window is even visible on the screen i.e. it may be covered up.	17
		No way to determine availability.	24

58	Not obvious who is attending the conference or if somebody has left without explicitly attending to another window. (m)	Couldn't figure out who was logged on without some fiddling around. (Needs instructions - maybe tool tips or balloon help.)	2
		Could not indicate to other participant that I was exiting. (Combine participant screen with drawing screen.)	2
		Don't know who is around (quickly at a glance). Easy enough to find that information and contact partner but not obvious that he's around at a glance.	3
		Have to highlight conference name to see users. None are highlighted by default so it was hard to know who is in what conference.	7
		Can't tell at a glance if others are present unless their pointer is on the work area. It is possible to look on the conference list but that is not at a glance.	7
		Having to ask "Do you see me?". Can't see the person - don't know off the bat who's there. Have to go to Collaboration to see participants. Still don't know where they are. (Have a visible list of who is logged on and where they are)	9
		Can't tell when the user is not working any more in the screen unless they tell you.	9
		The only thing I knew is that the person was there. Only after I chose the Participants option did I see that they existed. (The Participants dialogue should be up by default).	10
		I was unable to see who all the people in the Open Registration were without selecting each conference individually. (Have a show all participants)	10
		When my partner finally joined the new conference, there was no notification saying that he joined. Nor was there notification when he left. (Maybe a message saying that the person was there or a picture representing the person pops up).	10
		Place view of users in each room in main window, not separate pop-up window. This window easily gets lost in window clutter. (Add users' pictures to main window)	12
		While there is a way to see who has a GroupKit session manager open, I suspect that people will rarely raise it and therefore will not know. Also the open reg. is large, people will rarely have it running all the time.	17
		Unless participants window visible, can't tell who is logged on or off, or when person logged on or off. (Audio cues for log-on-off events.)	18
		I found it easy to find a conference, but sometimes I couldn't tell there was anyone in it. (Update automatically when user arrives.)	21
		No way to tell who's been around. If you want to see if someone's come and gone, there's no way to see.	21
59	People must explicitly start a conference in order to invite new members. This is heavy-weight even if an encounter is desired. (m)	People must explicitly start/enter a GroupKit conference, which makes it heavyweight even if an encounter is desired/established.	17
60	Cumbersome setting up the registry for a GroupDraw session. Requires coordination and communication in order to notify everybody of the registry and conference name. (m)	It is a little bit complicated using the program for the first time, and trying to set up a conference and get other users to come.	5
		People have to coordinate and set up a server before hand.	7
		Inviting a user to a conference - have to tell them what conference and where to go. (Allow for a direct invitation into a conference)	7
		People have to know the server name, there is no search for it.	7
		You have to know the host name to start the collaboration. You can't even pick one from a list and there's no obvious association between people and hosts. (Show a lot of available hosts associated with names of the people on the host)	8
		You have to know where a GroupKit is running to start collaboration. (Generally, server locations are probably published. Maybe search local subnet for instances or startup?)	21

61	Difficult to differentiate between multiple GroupDraw sessions. (m)	When connecting to system and both people create GroupDraw sessions how do you know which one to enter and which one is which.	9
		Double-click on the conference brings up new window and have multiple copies of self - one for each window and can't tell who each one is.	9
62	All participant windows look the same regardless of the conference. (m)	"Participants" window - how do you know if it's the participants of the GroupDraw conference or the conference for some other program without jumping to yet another window?	1
		Participant windows pop up separately for each room I subscribe to. If I move these windows to another part of the screen, I easily forget which room they were referring to. (Participant list should be built into the same room window, which they are referring to.)	4
		All the 'participants' windows look the same. Which chat belongs to which window? (Merge chat interfaces)	21

OS	Out of Scope	Titles of the 2 listboxes in the GroupKit window, "Conferences" and "Users" appear to be menu titles for the window. (Use white space or some other means to differentiate them.)	1
		Can't highlight or move continuous regions of the workspace.	2
		No facility to draw continuous lines - must only draw squares, circles, straight lines or text.	2
		Can't save your workspace.	2
		Found a bug … I could move a line and draw a new one at the same time. Very confusing for other participant.	2
		Clicking once on the drawpad while toolbar is locked produces a "square" that could be mistaken for another telepointer.	2
		Open registration - when type in "You are:" and your name, it is not clear what the program does because cursor still flashes in the same place after you press enter. As well, name doesn't seem like it is being processed. No feedback much less feedthrough. (Clear the text field once the user presses 'enter' and display somewhere that their on-line GroupDraw name has changed.)	3
		There is no way of saving the state of a room for a future session. In this way it is not possible to leave notes in a room suggesting a future meeting time - notes which other users can see when they enter the room at a later time.	4
		Regarding modifications of text boxes by two users at the same time, we spent too much time trying to figure out "how the system thinks". (If a simpler model is used, then users won't waste time trying to figure out its intricacies)	5
		Can't select region to manipulate.	9
		Can't save work to use later.	9
		No curved lines.	9
		Rooms vanish if no user in them, can't leave a note or message for others to meet you there later. (Keep a chat program running or send e-mail)	12
		List of users for system looks the same for list of users for a conference. (Keep them distinguishably separate on your screen or place a label on the window specifying which is for which)	12
		Didn't know to put my name in there. How do I submit my name? Do I need to? (Get user's name on login.)	21

NP	Not a problem (false positive)	Can't edit text once it's initially been entered into the canvas. (Allow text editing.)	1
		I could move and alter other's work without notifying them.	2
		Participants may erase each others drawings, which may lead to accidents.	6
		There seems to be no example of embodiment any where in this program.	6
		You can delete another person's work.	8
		When other person does something they draw and move lines around.	9
		Can't add text to existing ones - what if messed up or want to add to it?	9

NP	Cont'd	Text entered by other people (using the text tool) is not modifiable.	13
		Users must iterate through conferences to see who is online. (Provide a global list of users on-line)	16
		You can't tell who's connected to the GroupKit as a whole - you can only see on a per-conference basis. (Maybe add a "global" or "everyone" conference item as a default.)	21
		Can't message between sessions. If you want to page someone, you have to do it through the registrar.	21
		Can't see other sessions: if someone thinks to start a session, to join another one, it won't work: they will get their own session.	21
		Poor chat usability - can't tell who chat messages are sent to.	24

B.2 Mechanics of Collaboration heuristics – Groove

Table B.2 Groove Problems reported using Mechanics of Collaboration heuristics

Heuristic 1 – Provide the means for intentional and appropriate verbal communications			
1	Easy to miss a new chat message since there is no indication of its arrival. This is especially problematic if the chat tool is not open. (m)	It is impossible to tell if a new chat message arrives if the chat window isn't open. (Provide a visual indicator beside the chat tool.)	1
		Bad communication without microphones, only text chat available. If I am concentrating on something else, don't notice when partner is asking for help.	3
		Chat messages do not grab the person's attention. If he/she is not concentrating on the chat box, messages go unanswered. Others cannot ask for help if chat is used as the main source of communication.	7
		Chat tool - if the other person does not have the chat window activated/open, he/she is not aware of any messages and does not get any notification of message. (Provide an indication that messages are going on without interrupting the user. That way, the user can remain in his private space if desired.)	7
		If one person does not have the chat pane open, they don't know if another is trying to communicate with them.	17
		Cannot tell if others saw my chat message - could hide window. (Don't allow chat area to be obscured. Use additional feedback mechanisms (e.g. audio) when chat message posted.)	18
		No way to tell if someone has text chat up. (Should be able to tell! Someone might miss a message directed at them)	22
		Chat - difficult to notice that new messages have arrived.	23
		Easy to overlook incoming messages - text chat.	24
		No audio cue or other cue when text shows up in chat window.	25
2	Text conversation does not always appear in the correct order. (m)	Chat box - cannot see if someone is about to type or not, making conversational turn taking difficult.	7
		Using the text chat feature is annoying since you can't tell if anyone else is about to send a message, so conversations get jumbled. People started using the Notepad since it is at least pseudo real-time.	8
		Text chat message don't always appear in correct order. Makes conversation confusing. Someone may experience lag or hit send too slow. (Ask other for clarification - social protocol)	12
		Can't tell when someone is composing chat messages - good feedback for if they saw my previous message.	18
3	Chat messages can scroll off before an individual has a chance to read it. (m)	Chat box - messages can scroll off before people read it.	7
4	Hard to understand some utterances and the tone of voice due to the poor quality of the audio link. (M)	The spoken communication uses a low bandwidth method, which results in a low quality communication channel. (Boost the bandwidth to improve the quality.)	5
		When the microphone is locked down during conference mode, there is non-stop static. This gets so annoying that the users will be tempted to disable it.	5
		Similar to the above content, do not know who is talking due to the quality of the audio.	6
		The voice chat is really *low quality* and has a big delay.	8
		Really bad crackling.	9
		Can hear background noise - distracting.	9
		Delay in time (1-2 seconds) for voice. *Although there was a communication line through the voice channel, it was a poor quality. Text channel also limited expression and tone of message* (Work around the problem by using symbols in text chat to facilitate expression or tone, also speaking slower and waiting for the other to talk.)	12

4	Cont'd	The voice transfer with 4 people is really bad. Not only is the message delayed but with 4 users *two of them had horrible voice transfers*. (Better voice transfer would help communicate, since the chat box is sometimes not shown at start-up and could potentially leave someone out of the session as far as input goes.)	14
		Poor sound quality and large delays. (Use the phone - speaker phone)	16
		Similarly, the voice delay and *low quality* means that it is poorly matched to people's actions.	17
		Voice chat is poor quality - hard to understand some utterances.	23
		Verbal chat - *audio quality very poor*, long delays disorienting.	25
5	There is a delay between one participant speaking and another hearing the words. (M)	The delay in voice is extremely distracting. No sound would be preferable.	2
		Big delay on chat.	3
		There is considerable audio lag, which causes awkward breaks in communication.	4
		The time required for a talk message to transmit can reach up to 10min, making it hard to coordinate on-screen actions and verbal exchanges	7
		The voice chat is really low quality and has a *big delay*.	8
		Lag in voice very annoying - delay bigger with more people.	9
		Talking function lets everyone talk at once with the lag don't know if someone else is talking.	9
		Delay in time (1-2 seconds) for voice. Although there was a communication line through the voice channel, it was a poor quality. Text channel also limited expression and tone of message (Work around the problem by using symbols in text chat to facilitate expression or tone, also speaking slower and waiting for the other to talk.)	12
		The voice transfer with 4 people is really bad. Not only is *the message delayed* but with 4 users two of them had horrible voice transfers. (Better voice transfer would help communicate, since the chat box is sometimes not shown at start-up and could potentially leave someone out of the session as far as input goes.)	14
		Yuck. Voice is there but even on my home LAN the voice lag was ~0.5sec. Totally unacceptable. And what's with the "push-to-talk" button? Most audio cards support full-duplex audio. (Enable full-duplex audio and get rid of the "walkie-talkie" interface.)	15
		Poor sound quality and *large delays*. (Use the phone - speaker phone)	16
		Similarly, *the voice delay* and low quality means that it is poorly matched to people's actions.	17
		Audio delay makes comprehension difficult; words not well synchronized with actions.	23
		Verbal chat - audio quality very poor, *long delays disorienting*.	25
		Voice communication using the means provided is terrible: *long latency*, distorted and awkward interface to establish a voice conference.	26
6	Verbal communication portion of the interface is confusing, heavyweight and unreliable. (m)	There are two modes of communication - 'conference' and 'not conference'. It is not immediately clear that this really means 'full duplex' and 'half duplex'.	5
		Conversation - There is a beep to indicate that talk is on, but there is no indication of it being off. That means it is hard to know if one is connected for a conversation or not. One could inadvertently turn it off without knowing. (Provide a beep or turn down the static when off.)	7
		Can't hear anything and the "Hold to Talk" icon changes to grey - what does this mean?	9
		I have to press the button to talk, every time I want to say something I have to leave the workspace.	10
		Have to test and ask whether the voice channel is working - "Can you hear me?"	12
		Yuck. Voice is there but even on my home LAN the voice lag was ~0.5sec. Totally unacceptable. *And what's with the "push-to-talk" button? Most audio cards support full-duplex audio.* (Enable full-duplex audio and get rid of the "walkie-talkie" interface.)	15
		The 'hold-to-talk' did not work all the time - cannot tell if other people are getting the message or not.	17

6	Cont'd	Provides the means for intentional verbal communication, but it is very hard to use.	21
		The 'push-to-talk' method is too heavyweight for this type of communication.	23
		Voice communication using the means provided is terrible: long latency, *distorted and awkward interface to establish a voice conference*.	26
		Problem - 'push-to-talk' button was inconvenient; you could either talk or work. We wound up using the 'chat' mode for communication.	27
7	Audio tones are loud and displeasing. (m)	Sound effects are loud - displeasing to the ear; resulted in turning down the volume so that another person's voice is almost inaudible.	1
		The squelch from the walkie-talkie hurts the ears, even at the lowest non-muted setting. Participants may lose information from the speaker initially as they try to adjust their listening. (Change it to a more comfortable tone.)	6
		Big beep bad - hurts!	9
		Beep when talk enable is sooo annoying. Why beep when I can see that I pushed the button. (No beep!)	10
		Loud sound effects make it so that we didn't use voice chat - we were a bit hesitant after the initial burst and the text chat was sufficient.	13
8	No indication of who has the ability (hardware and drivers) to talk and who does not. (m)	Don't know who has the ability to talk unless you hear their voice.	9
9	Actions are not always coupled to the verbal exchanges. (m)	The delay and slowness in typing (in the text tool) means that voice cannot be tied to actions.	17
10	No feedback to tell if other person has heard an audio message unless they reply. (m)	When using talk, don't know if people can hear me.	9
		If a person is not active in a space (but it is still open) and another user speaks over the voice channel, it is not heard. There is no way to know a user is transmitting in this case or that the end user is not receiving. (Show focus of a user so others know if they are able to receive voice transmissions.)	11

Heuristic 2 – Provide the means for intentional and appropriate gestural communication			
11	Unable to perform intentional gestures. (M)	Despite the 'Navigate Together' option, it is impossible to tell what the other person/people are looking at ("this?"). (Provide a cursor to show where other people are pointing to.)	1
		No telepointer - no way of illustrating or pointing at things. (Provide telepointer.)	3
		Bad collaboration - if I want to tell partner to click on something, cannot gesture where. If I tell him over the mike, big delay and things are misunderstood. (Provide telepointer.)	3
		There is no way for one user to point to an object on the screen while conversing with another user. This can be really awkward because users must verbally describe the object they are referring to - a difficult task if there are many objects on the screen - rather than being able to say "this object" or "that object". (Each user should have a unique cursor, visible in real-time to other users. In this way, users can point to objects during the conversation. Cursors could even change shape to immitate the desired gesture e.g. extended finger for pointing to a particular object, open palm for referring to a broader region.)	4
		There is almost no indication what so ever that another user is showing you something on screen. It seems that they assume verbal communication will make up the slack in this respect. (Put in telepointers at the very least!)	5
		There is no means for gestural communication. (Provide a telepointer.)	7
		There is no obvious way of communicating gestural information. (Provide telepointers.)	8
		No way to highlight an area to bring to attention to the other users.	9
		There is no way to show the other person what I am doing. Can't say/point "look at this"	10
		I could not see the other user's gestures as there was no telepointer to show what they were doing. (Show telepointers of each user on screen.)	11

11	Cont'd	*No intentional gestures i.e. can't point to anything, no deixis. Must say "I am working on the second cell from the top" or "look at the second one".* Can't see nods or acknowledgements by bodily actions. (Telepointer to show his actions or gestures. Visible cursor during text typing. Video to see facial gestures or bodily actions.)	12
		Voice and text chat allows people to refer to objects in the workspace, but does not provide the means for deictic references - e.g. I could say 'this row' but no one would know what I was talking about. (Provide telepointers)	13
		Users are unable to see the gestures of others through mouse movement or cursor changes. (Makes it difficult to communicate your intent for a task. Add the ability to change your mouse to a pointer for gestural movements and to allow people to see where you are working.)	14
		Intentional and appropriate gestural communication completely unsupported. I can't see other person's cursor. I can't see what they've drawn or typed until it's finished.	15
		There is no support for gestural communication.	16
		There is no embodiment in Outliner, no telepointer, no indication of who is doing what, etc. *No ability to gesture.* (Provide telepointers.)	17
		Because there are no facilities for visual gesturing, this important requirement (provide the means for intentional gestural communication) is not supported at all.	17
		Cannot see what people are pointing at when they use deictic references in text/voice chat messages. (Provide telepointers.)	18
		No means for intentional gestural communication. Not even 'inisde' the system such as telepointers when drawing. Only see the final result.	19
		No facility for gestures. Cannot see what others are pointing to. (Provide telepointers.)	20
		No suport for providing the means to do intentional gestures.	21
		There is no way to point to things. (Provide a telepointer)	22
		No telepointers - no ability to gesture or point.	23
		No way to point to objects in shared workspace. No way to tell what objects other users are examining.	24
		No support for gestures, no telepointers or means of seeing others on the screen.	25
		No means for gestural communication that I found.	26
		No gestural communication that I could find.	27

Heuristic 3 – Provide the consequential communication of an individual's embodiment			
12	No indication of what somebody is currently doing. (M)	Confusing to figure out who is doing what for Outliner. A button expands/collapses the information about who posted text. Didn't notice this button for quite a while.	2
		Cannot tell what other person is doing his actions. (Provide telepointer.)	3
		In general things shift around without me knowing why!! Eva demoted something and my view instantly collapsed!? At time, did not know what she did.	3
		There is very limited indication of where somebody is or *what they are doing.* The only small bit of info they do provide is which toolspace the other user (and yourself) are in.	5
		There is no embodiment, no feedthrough when one participant switches context. (Give good indication of the embodiment.)	6
		You can't tell when people are using the various tools in the Outliner, Sketchpad, etc.	8
		Not aware of what other people are doing in relation to yourself.	9
		Don't know who's in the Outliner - found tooltip on tab but still don't know who's writing what.	9
		I had no idea what my partner was doing i.e. I did not know where my partner was working or what he had on his screen. (Context view of partner's position.)	10

12	Cont'd	*Don't know if he is thinking, working on something or has left his computer, or even sleeping.* No bodily actions are available to understand what is happening. Don't know what he's looking at. (Small video display to show his facial gestures. Better audio channel to receive clear speech dynamics.)	12
		When we were both using Outliner, I couldn't tell what he was doing i.e. what is he looking at, where will he go next. (Provide telepointers for the actual work window)	13
		There is no embodiment in the outliner, no telepointer, *no indication of who is doing what, etc.* No ability to gesture. (Provide telepointers at a minimum.)	17
		No location, gaze, view, or *action support*.	19
		No indication of others cursor. Don't know what people are doing, what objects people are holding.	19
		Difficult to determine what the other person is working on. Difficult to tell whether other person sees the same view (items) as me, so hard to divide the task.	23
		No way to tell what mode a user is in i.e. typing an entry, modifying fonts, etc.	24
		No idea what others are currently doing from looking at screen.	25
		No real indication of what the other people were doing or were going to do - at least within the Outliner, you could tell that somebody was in the Outliner but that was it.	27
13	No indication of where somebody is located within the Outliner tool. (M)	No embodiment - telepointer or something cuz don't know where person is. (Provide telepointer)	3
		Cannot tell location of other person.	3
		There is no indication of where the other user is currently working in the workspace. It is difficult to determine if my partner is actually working on a section of the Outliner, or if he is merely taking a break. (Display the cursor of each user in the virtual workspace. Movements of the cursor and actions performed should be updated in real-time.)	4
		There is very limited indication of where somebody is or what they are doing. The only small bit of info they do provide is which toolspace the other user (and yourself) are in.	5
		Very confusing what is happening and where people are.	9
		Can't tell where the person is looking or *manipulating*.	9
		I had no idea what my partner was doing i.e. *I did not know where my partner was working* or what he had on his screen. (Context view of partner's position.)	10
		I had no idea where the other person's pointer was. (Show the other pointer)	10
		Difficult to tell which area of the space each user is in at a glance. (In the user list display the user's current focus beside their name.)	11
		Unclear where he is working. Hard to help each other or not work on the other's area. Appears as individual work showing up on a joint space. (Show what the other person is doing while they do it. Use relaxed WYSIWIS so easy to follow the other's actions. Provide telepointers. Have better audio channel to enable social methods of collaborating.)	12
		No location, gaze, view, or action *support*.	19
		No support for gestures, *no telepointers or means of seeing others on the screen*.	25
		No telepointers - impossible to tell where others are.	26
14	Cannot tell which part of the workspace a person is looking at (no gaze awareness). (m)	*Can't tell where the person is looking* or manipulating.	9
		Don't know if he is thinking, working on something or has left his computer, or even sleeping. No bodily actions are available to understand what is happening. *Don't know what he's looking at.* (Small video display to show his facial gestures. Better audio channel to receive clear speech dynamics.)	12
		When we were both using Outliner, I couldn't tell what he was doing i.e. *what is he looking at*, where will he go next. (Provide telepointers for the actual work window)	13
		No location, *gaze, view*, or action *support*.	19
		No way to point to objects in shared workspace. *No way to tell what objects other users are examining*.	24

15	Difficult to keep track of individuals as they navigate to different tools in the space. (m)	Changing to different areas like drawing to brainstorming give too little indication to other participants. It does give a small tooltip when one does change, but it goes away quite quickly thus easily missed. (Show the "location" of participants i.e. the tool being used on the user list)	6
		The workspace tab has a numerical indication of how many participants are in that particular shared workspace. However, that required figuring out initially. Quite misleading as it led us to believe that it indicated how many windows were opened or something to that effect. (Use iconic representations - use pictures of people...)	6
		When you're using a different area (e.g. Sketchpad, Outliner, etc.) than others, the only info on what they are doing is the number of people in that area - no immediate indication of who they are or what they are doing.	8
		Tool tips along bottom to see where everyone is going to - not clear and annoying.	9
		Not clear where people are if you don't follow/remember what the tool tips said.	9
		The transitions between spaces are unclear. The number on each tab only shows how many, but not who. If I am working in my own space (which I can't be sure is my own) I can't share without difficulty. Coordination without chat and/or voice would be nearly impossible.	11
		Tooltips "so and so has entered …" don't disappear fast enough, so it is possible for several such tips for the same person to simultaneously be on the screen.	18
		Cannot see in a persistent fashion who is in which tab unless you mouse over it. (Icons for people arranged in tab labels.)	18
		Difficult to quickly see who is online and actively working on a tab workspace. (More straightforward display of who is active in a tab with you - not requiring you to mouse over the tab name.)	20
16	Unable to observe visual cues (e.g. head nods) given off by collaborators. (M)	The audio channel alone does not provide a means to observe subtle cues given off by the speaker's face. There is little conversational awareness as a result. (Video might be an option.)	4
		There is no consequential communication of individual's embodiment at all. We cannot see anything from the other participants.	6
		There is no communication of embodiment.	7
		Since there is no way to communicate gestures intentionally, there is certainly no way to communicate information unintentionally.	8
		No intentional gestures i.e. can't point to anything, no deixis. Must say, "I am working on the second cell from the top" or "look at the second one". *Can't see nods or acknowledgements by bodily actions.* (Telepointers to show his actions or gestures. Visible cursor during text typing. Video to see facial gestures or bodily actions.)	12
		No embodiment what so ever.	15
		There is no embodiment of the user to give off unintentional communication.	16
		Because there is no embodiment, there is absolutely no opportunity for receiving consequential communication from it.	17
		Can't tell if person heard my audio message unless they reply - and the reply gives sufficient clues to decide. (Need conversational awareness channels.)	18
		No embodiment - so how can it provide consequential communication?	21
		No real embodiment - some actions are visible but not many.	23
17	Impossible to differentiate between several participants with the same name on the network. (m)	Allows you to have multiple identities - but how do you know which ones you used for different spaces also if people have same name/similar name.	9
		Possible for several people (instances) with same name on network - impossible to differentiate. (Use unique user IDs)	18
18	Confusing for participants to change their name/identity. (m)	Couldn't easily change my name/identity.	2
		It took some time to figure out how to change my account name in a space and to find out the 'computer name' of the other person since the default name is the computer name. (Allow for the changing of an account in a space instead of just in the general Groove section.)	7

18	Cont'd	I couldn't figure out how to change my name to be something other than my machine ID.	8
		Identity of individual is unclear. Host name of computer is not really a good way to know who everybody is - I'm sure there must be a way to use real names, but couldn't find it. (Video of individuals. Small picture of person be each name. Easily modifiable names.)	12

Heuristic 4 – Provide consequential communication of shared artifacts			
19	No indication of what somebody is doing until they are done. Intermediates steps of actions are not displayed on the remote sites. (M)	Can't tell if someone is performing operations until they are finished.	2
		No feedthrough - don't see what person is typing or doing. Articles just suddenly appear. (Make application real-time. See what other person is typing while he's typing.)	3
		Cannot see what other person is editing! No indication what other person is doing.	3
		Typing in a record is not real-time - the other person will not see what I am typing till I press 'enter'. This makes it difficult to determine what another person is currently doing - I might assume he is taking a break, when really he is typing a long entry into a text field. (Update the records in real-time. Display the text cursor positions of other users.)	4
		There is no real-time feedthrough of the modifications that another user is making to an object. (Provide real-time feedthrough)	5
		In the brainstorming workspace, participants cannot see the text typed until enter is pressed. So one can overwrite/delete another person's information without realizing it.	6
		No indication of someone *editing* or selecting a record. (Show intermittent steps)	7
		No warning of when the other user is modifying a record, surprise with its appearance. Others cannot coordinate because they cannot see that it is being changed. (Show that a record is being modified while it is being modified.)	7
		When lines are added to the Outliner, don't know what person is writing, you just see it appear.	9
		No way to tell if an action is in progress. Only see a new artifact or change after it has been completed. This is especially troublesome when one user is modifying an object being changed by another user. (Provide intermediate feedback of user actions to all participants.)	11
		Can't see cell typing when it happens or other object manipulation like changing hierarchy levels of cell. Just see final position making it unclear as to what happened. No way to see its history or how it got to the new state. (Would require too much verbal or text communication to overcome easily. Action indicators to see the other user manipulating the object. Telepointers or real-time viewing of their actions>)	12
		Editing can occur simultaneously - last one to finish keeps the changes. *No way of seeing if a person starts editing a field.* (Show others when someone starts editing a field to help with social protocols or control who can edit a field - provide a first-come-first-serve technique.)	13
		No user actions are visible until completion - specifically, users cannot see other users mouse gestures, because they are not visible. (Show the other users mouse and cursors in a different colour so that all users understand each other's tasks and gestures.)	14
		Intentional and appropriate gestural communication completely unsupported. I can't see other person's cursor. *I can't see what they've drawn or typed until it's finished.*	15
		There is no feedthrough of object manipulation - only once it is complete. (One does not know who moved an object without verbally asking)	16
		We don't see items in the Outliner appear until after the line has been edited. (Provide immediate editing)	17
		The Outliner fails here (provide consequential communication of shared artifacts) in almost all ways because there is almost no feedthrough except for discrete steps e.g. like entering a completed line.	17
		In Outline tool, there is no indication about another person editing an item. (Show the incremental edits of other users.)	20
		There is no way to see what someone is adding while they're adding it. It just appears. (Character by character editing.)	22

19	Cont'd	Actions only show up after return is pressed; difficult to tell actions as they are happening.	23
		Difficult to tell what other users are doing until their action has been completed.	24
		No intermediate information available from artifact manipulation.	25
20	No indication on the remote participant's screen that an item has been selected. (M)	Users can type in the same record at the same time, overwriting what the other user is typing. This is very problematic because *there is no way to visually determine what field the user is currently typing into.* (Show typing in real-time - this way I can tell where my partner is editing text, and I know I should keep clear. Fields which are currently being edited could also change colour. Alternatively, fields could be locked to prevent more than one user from editing at once, but this method could also be too restrictive.)	4
		Very poor support of consequential communication of shared artifacts. There is no indication when another user is making modifications to a particular object, or even if they have selected it.	5
		In the shared workspace, when a participant wanted to highlight a point, he/she would normally select the text with the mouse. However, it does not display the selection on other screens.	6
		No indication of someone editing or *selecting a record.* (Show intermittent steps)	7
21	Changes to cells are too subtle. Cannot tell when something changes if not looking directly at it. (m)	My partner deleted an object from the view and I did not know where the object went.	10
		Cannot tell when something changes, e.g. participant enters, items added or modified, if I am not looking directly at it. (Audio cues to indicate changes are happening; change awareness features.)	18
22	Cells are the same regardless of their history, who created them, or who is currently working on them. (m)	When moving text cells in Outline, it is difficult for other people to locate and identify what cells have changed - same thing when you change the text in a cell. (Provide a visual indicator that will stay visible until the user acknowledges that the change has occurred; use animation.)	1
		No history of text - can't tell what an old copy looks like.	9
		The details view was inadequate, I had no idea who the last person to modify the object was.	10
		Can't see cell typing when it happens or other object manipulation like changing hierarchy levels of cell. Just see final position making it unclear as to what happened. *No way to see its history or how it got to the new state.* (Would require too much verbal or text communication to overcome easily. Action indicators to see the other user manipulating the object. Telepointers or real-time viewing of their actions>)	12
		Unclear of who was working on something by looking at it. Not sure if the object is finished or if work is even presently being done to it. Can't see last person that modified if without clicking show details, but this limits screen space when visible. (Can ask in voice channel for assistance or ask who was doing what - time consuming. Represent object with darker colour the more it is modified. Place small initials in corner of object for last modifier.)	12
		Users entering a task can see (through an option) which user has created/modified a task and the date/time that this occurred. It might be nice to include icons for each user and that icon is displayed beside each new item added to give instant interpretations for other users. The other details are nice but when not shown (took a while to figure out) users are unsure who is adding what.	14
23	If a view of a list is collapsed while another participant expands the list and adds/edits/deletes an item within it, there is no indication that this action has occurred on the other's display. (m)	If you add a cell and another user's tree is collapsed, they probably won't realize that something has been added - same thing goes with deleting. (Provide an icon beside a new cell that will remain visible until user acknowledges it and/or an audio indicator.)	1
		Don't have same views. If I collapse my tree, Eva doesn't see that. Her view is still expanded - gaze and view problem.	3
		My partner collapsed a tree view and the tree in my view stayed the same.	10
		Due to having different views, you no longer can see others adding to a list that is collapsed in your view.	13
		The individual views means that we don't see the same thing i.e. I can have mine collapsed but I don't know the other person has it expanded.	17

23	Cont'd	When Outliner items are collapsed, it is not easy to see if some of the hidden items have been changed or are changing. (Make tabs that have sub-items changed bold, as is done for e-mail/newsgroup programs.)	20
		When you collapse record hierarchies, other people can't tell you have done it. Hard to reference something if you can't see it.	22

Heuristic 5 - Provide protection			
24	Two participants can simultaneously access and edit the same cell without any indication of the other. Only the changes from the last person to hit 'Enter' are saved. (M)	If multiple people edit the same cell, the last person to finish editing and press 'enter' will have his/her text as the updated version; the first person's text is lost. (Lock the cell once someone starts editing it and provide an indicator beside it until the editing has finished.)	1
		Protection - lets 2 people manipulate same article at the same time.	3
		I don't see changes when I'm in editing mode. So if we both edit the same cell, the change that predominates is accorded to whomever pressed 'enter' last.	3
		Users can type in the same record at the same time, overwriting what the other user is typing. This is very problematic because there is no way to visually determine what field the user is currently typing into. (Show typing in real-time - this way I can tell where my partner is editing text, and I know I should keep clear. Fields which are currently being edited could also change colour. Alternatively, fields could be locked to prevent more than one user from editing at once, but this method could also be too restrictive.)	4
		There doesn't appear to be any protection what so ever! Due to the lack of feedthrough we can't tell that another person is writing in a record, so if we start writing there too, we can totally obliterate their text. (Provide some technical protection if they refuse to give any real feedthrough.)	5
		A person can change another's record. More than 1 person can be in editing mode at the same time. If one person is in edit mode, he/she does not see any changes that another person is making at the same time. Changes overwrite others that are being made at the same time. (Show that changes are being made and allow changes to take effect immediately.)	7
		People are free to delete/*change* other's work in the shared spaces.	8
		Can write on the same line - frustrating when trying to do own work.	9
		My partner and I edited the same object, I finished first and when he finished, all my changes were gone.	10
		In Outliner, users can change one author's entries without any kind of notification to the owner- undo function only works for local actions. (Allow users to lock their own items or to choose if it is an open item, accessible to all. Make undo functions to encompass all actions - especially when working with artifacts belonging to other users).	11
		Can delete others' message in hierarchy by clicking trashcan or typing something new. *Simultaneous typing or interaction with an object is not really supported.* (Show what the other user is doing i.e. telepointer or real-time display of his actions. Social methods could then overcome this problem.)	12
		Editing can occur simultaneously - last one to finish keeps the changes. No way of seeing if a person starts editing a field. (Show others when someone starts editing a field to help with social protocols or control who can edit a field - provide a first-come-first-serve technique.)	13
		I can edit an apparently empty cell that another user created. My text will remain and their changes will be lost. I can also delete the record while the other is still editing. (Provide feedthrough. Social rules should provide protection.)	16
		One person can edit a field. Then another person can edit the same field. Last one to enter it 'wins'. But there is no indication that multiple people are editing the same thing.	17
		Two people can edit the same item in the Outliner tool at the same time. Neither person is aware of the other.	20
		No protection for records. If someone edits something while someone else is editing, whoever finishes first will lose all their modifications when the second finishes.	22
		No ownership of entries, it is easy to over-write another user's work.	24

24	Cont'd	People can modify same item at same time without knowing, no indication that this has happened.	25
		Other users can over-write text that was not yet passed into the system by clicking 'Enter'.	26
		Didn't seem to be any protection.	27
25	A participant can delete any cell (or parts of it) even while another is editing it. (M)	Similar to above scenario, but someone can actually delete a cell when someone is editing it, in case it is removed immediately, leaving the person who was editing it wondering what happened. (Preventing people from deleting it.)	1
		I could erase other entries.	2
		If one person is editing a cell while another person deletes that same cell, person one ends up editing the cell above the intended one - edits the wrong cell cuz the one intended to edit was trashed!	3
		Another user can delete your information before you are done with it, or even delete an entire toolspace without your permission. (Require that both people give consent before a toolspace is deleted.)	5
		People are free to *delete*/change other's work in the shared spaces.	8
		Can delete other stuff.	9
		Can delete other's work.	9
		While my partner was editing a record/object I deleted the object and his editing was gone.	10
		Can delete others' message in hierarchy by clicking trashcan or typing something new. Simultaneous typing or interaction with an object is not really supported. (Show what the other user is doing i.e. telepointer or real-time display of his actions. Social methods could then overcome this problem.)	12
		Other people can delete rows that you are currently editing. (Provide some means to show when a field is being edited e.g. show the editing as it is occurring)	13
		No protection is given on the individual's work as others can delete the work they have done at any time. (Items created by users can only be deleted by the same user.)	14
		I can edit an apparently empty cell that another user created. My text will remain and their changes will be lost. *I can also delete the record while the other is still editing.* (Provide feedthrough. Social rules should provide protection.)	16
		If one person is editing a field as another person deletes it, it disappears on them. (Make actions visible so one can know another's intentions; make deletes in these cases not allowable.)	17
		If others delete items while they are being edited, there is no indication to the editor that someone deleted it - it just disappears. (Don't allow others to delete things while they are actively being edited. Have deleted items fade away gradually.)	18
		If someone deletes a record while I am editing, my editing disappears.	22
		Anyone can delete anything; there is no control over what gets removed. (Provide mark for completion or need 2 people to concur.)	22
		People can select and delete others letters within an item.	23
		People can delete items without warning to others … just disappears from screen.	25
26	A participant can move a cell while another is making changes to it. (m)	If I attempt to move the cell that Eva is editing, I don't see her edit the cell but the cell moves around on her.	3
		As I edit a cell, the other user can reposition the cell within the Outliner hierarchy. My cell jumps around on the screen, but I have no idea who is moving it. (Display an outline of the cell as it is being moved around. Also display a unique cursor, which is moving this cell - so I can associate it with the user who is performing the action.)	4
		It is possible to move a record that someone else is editing. (Show that a record is being edited/moved so that social protocols can be used.)	7
		Demoting and promoting cells - if someone is typing, their cell will move in real-time to new location. They don't know why. (Social protocols - WYSIWIS: show what other users are doing so people can intentionally not perform aggravating tasks to other users. Technological protocols - access control for movement of cell features or provide undo capability)	12

27	Unable to use social protocols to provide protection. (M)	The lack of feedthrough and technological protocols forces a reliance on the audio channel and social protocols - which suffer due to the annoying static on the audio channel.	5
		Providing of protection is not adequate. Two people editing an item, who finishes 1st and hits 'enter' wins and the other loses what he was typing. *No support for using social protocols.*	19
		The 'details' view is not very detailed. Why isn't there a 'status: modifying' or something. This would eliminate some communication problems.	22
		No protection of items and not enough awareness information for social protocols.	23
28	Cannot resolve conflicting actions once they have occurred without deleting the actions and starting over. (i.e. no undo function) (m)	A user can easily delete a field without the other user's permission. There is no way to recover from such a deletion. (Implement undo functionality. A user who authors a text field could also have the option of locking the field to prevent deletion or he could place a marker on the field indicating to others that he wishes the field to remain unedited - although people can still edit/delete it, but they will know it is against the author's request.)	4
		There is no undo function. This is especially a problem if we have inadvertently overwritten their text because of the basic lack of protection. (Provide undo functionality, at least for things that your yourself have done.)	5
		Similar comment, because participants can cause accidents of overwriting/deleting ideas. An undo functionality would be desirable. However, this is not available. (Provide a trashcan/recycle bin functionality, so that participants may bring "scrapped" ideas backed.)	6
		In Outliner, users can change one author's entries without any kind of notification to the owner- *undo function only works for local actions*. (Allow users to lock their own items or to choose if it is an open item, accessible to all. Make undo functions to encompass all actions - especially when working with artifacts belonging to other users).	11
29	One can 'uninvite' another from the session without them knowing what is going on. (m)	One can 'uninvite' someone from a session without them knowing you are doing it.	17
		You can unexpectedly uninvite people from the space - they can't recover. (Don't allow this unless the person has been inactive for some time.)	18
		No protection - I just gave Saul the boot from a workspace. (Create multiple levels of access.)	21

Heuristic 6 – Provide management of tightly- and loosely- coupled collaboration			
30	Difficult to shift between loosely and tightly collaboration due to lack of awareness and feedthrough. (M)	If another user is requesting help over an audio or text channel, it may be difficult for me to determine where the user is working in the Outliner, especially if it spans several spaces. (A mini-outliner window would display the entire document and show the location of other users. By clicking on the mini-window, I am zoomed into the corresponding location on the main window.)	4
		There is no real tightly coupled collaboration due to a lack of feedthrough.	16
		While this tool does let me do individual work, it gives no way to move into tightly coupled work as cannot link actions.	17
		No overviews so you don't know in which area of a tool a person is.	19
		No built in functionality to facilitate the management from tightly to loosely coupled collaboration.	21
		Difficult to draw someone's attention to an outline item in order for them to shift from loose to tight collaboration.	23
31	People can individually navigate within and between tools; however, there is no guarantee that everybody will see the changes since they may not be looking at the same portion of the workspace. (m)	With Chat window open, the Outline tool window requires a lot of scrolling once the tree gets full - people may not end up looking at the same area.	1
		If the Outliner grew in size and I am working on one section, I cannot see changes being made in another section. In fact, I wouldn't even be aware of some brainstorming session that is taking place at the bottom of the Outliner, if I am working near the top or middle. (Display a miniature version of the Outliner which displays the entire document as well as where activity is taking place. I can click anywhere on this mini-version to zoom in on the location - allowing me to quickly navigate to areas where others are working and may be requiring assistance.)	4

32	The 'Navigate Together' function creates confusion amongst users as they leap from tool to tool out of their control and not certain who is in control. (m)	Our session had 4 people - couldn't find them sometimes. Arbitrary switching between workspaces happened - we couldn't explain it.	2
		If navigating together, if one person is attempting to add an entry while other person uses the text chat facility, text chat overwrites the item person is trying to insert. If no microphone available, this makes communication horrendous.	3
		There is a "Navigate Together" toggle, which seems to be designed for enabling or disabling tightly coupled collaboration. Currently, it does not perform any function and now there is no private space where I can work on my own without disruptions from my collaborators.	4
		Clicked Web Browser and nothing is on it. Does it work? Clicked 'Navigate Together' - don't know what is happening.	9
		When navigating together, you don't know who is in charge or if everyone can do stuff.	9
		Navigate together button doesn't seem to provide any functionality.	12
		Although the Navigate together button is a great idea, it leaves a control freak door open. If someone takes control of the work being done by changing the task they are working on (i.e. change to notepad, sketch, etc.) all users are shifted. This is frustrating if someone is working on something and is transferred to another area because of a user selection. (Perhaps another window (smaller version) could be shown to show where the group is working while you are completing your task.)	14
		If one selects the 'Navigate together' option, the last person to click the button forces the navigation to their page. Therefore, if you want to join others, your act of 'navigate together' actually jumps to your current tool, which is the wrong effect. (Should have an idea of 'master' navigator or 'join'.)	17
		Navigate together' can cause loss of keyboard focus or loss of entered, but uncommitted text.	18
		'Follow others' function stinks - I pressed it thinking that I would navigate with the others; instead they were suddenly forced to for where I was.	21
		I have no idea what 'Navigate together' does.	22
33	Difficult to establish a private space since there is no indication of what area of the workspace people are working on or looking at. (m)	There is a "Navigate Together" toggle which seems to be designed for enabling or disabling tightly coupled collaboration. Currently, it does not perform any function and now *there is no private space where I can work on my own without disruptions from my collaborators*.	4
		No private area to work from what we found. So everyone can see what everyone else is doing in the same space.	6
		Can't tell if a person is working in their own or if they are in the process of adding/manipulating shared workspace.	9
		Unclear where he is working. *Hard to help each other or not work on the other's area. Appears as individual work showing up on a joint space*. (Show what the other person is doing while they do it. Use relaxed WYSIWIS so easy to follow the other's actions. Provide telepointers. Have better audio channel to enable social methods of collaborating.)	12
		Everything is private when I am creating it, but public when I am done creating it. I can't easily work on my own without showing an object until it is completely done. I can't easily collaborate with others during the creation of an object. (Allow multi-user access to manipulation of objects. Show telepointers and let them access the same object. Provide overviews of space and separate work places as well as combined ones.)	12
		Users are unable to create their own set of tasks without them being added to the "groups" set of tasks. (Add the ability to create a disconnected set of tasks that can be seen by other users but cannot be changed/added to. When the user is finished she/he can add that set of tasks to the group set.)	14
		Does not promote switching between levels of coupling. Instead, work is mostly individual with a shared view.	24
		Cannot control which portions of a document are shared. All items shared once 'return' is pressed rather than when toggled by the user.	26

Heuristic 7 - Allow people the ability to coordinate their actions			
34	Coordination is severely limited due to the audio delay and lack of feedthrough, embodiment, and protection. (M)	Coordination is severely limited especially due to the lack of feedthrough and protection. You have very little idea of what other people are doing, other than what they explicitly tell you on the audio channel.	5
		Since don't know where people are adding or manipulating things, hard to coordinate actions unless through talking but there is a huge delay.	9
		Since users cannot see other users' mouses, it is impossible for them to coordinate their tasks. If user A was adding a new item to the list then the other users not only don't see user A adding the new item but also adding text to that item. (Show the new items being added in real-time and then show all the information being filled in as users are typing)	14
		Since there is no visual feedback of any kind about what others might be doing, it's pretty hard to coordinate actions.	15
		Coordination is almost impossible because many of the previous mechanics are not supported.	17
		Poor support for social protocols to allow people to coordinate their actions. People only see 'final' result of actions so it is hard to coordinate.	19
		No built in functionality to help people to coordinate their actions.	21
		No tools to explicitly facilitate activity coordination.	24
		No coordination of actions is possible within this tool. (Use chat/telephone/e-mail/vocie over IP to communicate coordination of actions.)	26
35	Participants can independently change the ordering of the items so that it is no longer synchronized on different systems. (m)	His view was often different from my view when we collapsed lists or re-arranged row ordering e.g. doing it on my screen did not change his screen at all. (Give feedthrough as to what rows are being manipulated by others and the results of those actions - highlight those rows a different colour)	4
		It is possible for two people to sort the outline differently in Outliner. Thus, if my partner's is sorted alphabetically while mine remains unsorted, her gaze and view is completely different than mine. Thus, if she wants to change the 3rd item and tells me that, my interpretation would be completely wrong. (Either provide overview or force all collaborators to share the same view.)	17
		It is possible to sort the columns so the views may be different for each user. There is no indication to users that views could be different causing problems when users refer to different things.	17
		Difficult to determine what the other person is working on. *Difficult to tell whether other person sees the same view (items) as me, so hard to divide the task.*	23
36	Cannot anticipate actions of others. (m)	Couldn't tell if someone was about to add or edit an entry.	2
		Collaborators switched from workspace to workspace - couldn't tell intentions.	2
		Can't anticipate the activities of others at all unless they tell you.	9
		No way to see what a person is planning to do or where they are on the screen.	9
		No way to anticipate actions of others - makes it difficult to coordinate actions.	25

Heuristic 8 - Facilitate finding collaborators and establishing contact			
37	Difficult to determine who is around and their availability state when trying to start a shared space for collaboration. (m)	Cannot tell what other person is doing if he/she is working in a different space.	3
		There is no iconic indication for the person's availability state i.e. "I am free to read and respond to your messages" or "I do not want to be disturbed by any messages at the moment." (Place a door state icon beside each user's identity. The state of the door indicates how busy the user is.)	4

37	Cont'd	There is no notification of when someone else has logged on to Groove so it is difficult to find out when someone is available. The only way we found was to continually check the list of people. (Allow for a notification of availability for selectable contacts.)	7
		Someone who is running Groove cannot set some indication that they would not like to be disturbed. (Provide a status indication that users can set.)	7
		Not obvious where I am and who is around.	9
		No way of seeing if the person is off line just for that one session or if they are off-line for all.	9
		Have to go to a completely separate screen and leave workspace to look up where people are - too much work.	9
		It is difficult to find collaborators if you don't know who they are. It is also hard to tell if a known user is actually available to collaborate. (Create groups/rooms to make finding users easier. In addition to showing that a user is online, show if they are available to work.)	11
		Finding person and space confusing. It was hard to establish a workspace and get the other user in it. *Wasn't sure if they were even present on the system.* (Make sure list more accessible. Graphically show presence of others from your working domain.)	12
		Difficult to see if others are away or not currently running Groove - you can invite them but you won't get a reply.	13
		The user list does not really show who is around in other spaces - very confusing to know people's state.	17
		Too many windows to be able to send a message. *People are not required to fill out full name so it is hard to locate them on groove.net.* You always have to invite people so they can enter your spaces. People cannot see if I move to another application since with Groove open it shows I am on-line.	19
		No way of knowing whether someone is actively attending to a work tab or if he/she has switched to another application or left the computer. (Show user's focus and match idle time.)	20
		I had great trouble finding anyone!	21
		Confusing to determine what 'space' someone is in.	23
38	Difficult to start a new shared space and invite participants or join one in progress. (m)	I had a generally hard time getting started. Couldn't figure out how to join a workspace.	2
		The program was hard to set-up initially. Participants trying to start a collaboration made several requests to each other and ended up with multiple sessions some each in different sessions. Very confusing.	6
		You can't see who is in a space to go and join in with them.	8
		Asking people to new spaces becomes confusing if there is a lot happening.	9
		How do I find people from my new space? Very frustrating - hard to see invite button.	9
		When I first started Groove I got a message from someone I did not know asking me to join a workspace that I knew nothing about. (More details in message - sender, workspace)	10
		Finding person and space confusing. *It was hard to establish a workspace and get the other user in it.* Wasn't sure if they were even present on the system. (Make sure list more accessible. Graphically show presence of others from your working domain.)	12
		Establishing contact is confusing! A new window/session is opened while the existing session you were currently in remains open. There is a vague indication of which session you are in. (Perhaps the ability to have more than one session running in the same window and a better indication of which session you are currently in would help enormously.)	14
		Voice only works after people are in a workspace, but we found voice is needed to let people coordinate how they get into a workspace.	17

39	To join a shared space that has already been set-up, one has to be invited in. (m)	To join a space that is already been set up, you have to invite people - seems backwards. (Have a wizard-like thing that walks you through the process of hosting a session or joining one.)	8
		Too many windows to be able to send a message. People are not required to fill out full name so it is hard to locate them on groove.net. *You always have to invite people so they can enter your spaces.* People cannot see if I move to another application since with Groove open it shows I am on-line.	19
40	Not intuitive on how to invite multiple people at once with the instant messaging tool. (m)	Inviting a user, I could not invite multiple users - only one at a time. (Allow multi-select.)	10
		Have to invite everybody individually, why can't they just find room and ask to join. May be tedious to invite 20 people. (Allow others to join room without invites. Show listing of all rooms within your specific domain.)	12
41	When establishing contact, the message appearing in the bottom right corner of the screen is confusing and easily overlooked. (m)	Do not notice Instant Message - establishing contact hindered. (Sound when message is received.)	3
		The green pop-up window in the lower right corner of the screen contains a history of e-mail messages which users can send to each other. The history list grows in size as messages are added, but when I select a message to read it is deleted from the list and displayed in a separate window. If I bring up too many messages at once, it is easy to lose the sense of order in which they were sent because one can no longer see the order in the list. (Retain the message arrival order in the history list. Offer an explicit delete option for removing messages or simply push old messages down the list and out of sight. In this way communication through the e-mail facility retains a sense of order/sequence.)	4
		We used the messaging to contact each other but it was very easy for the messages to go unnoticed. We ended up not responding to each other as quickly as we would have in real life. (Make the status tray notification more noticeable.)	7
		How do you reply to the instant message?	9
		The invitation facility and response (by the green pop-up on the tool bar) is confusing - the several stacking status indicators appear and disappear. It took several of us talking face-to-face to make this all work.	17
		Invite' facility arrives outside workspace; hard to notice you have been invited.	23
42	Not immediately obvious who is actively working in the current space or when somebody leaves. (m)	If a person leaves the space, Groove does not update the list of members in the space. Thus a person who has left the space is still displayed as being present. (Update that list if a person leaves the space.)	3
		Groove displays members present in a room using their names. However, if there are many names on the list it may be difficult to determine "at a glance" who is around. (Instead, or in addition, to a name list allow users to display their portraits on the members' present list.)	4
		Can't tell that a user has left the space.	7
		While it lists people on-line, it is hard to tell by the green light where they really are i.e. if they are in or not. Also easy to miss their arrival.	17
		User left workspace and joined another. I couldn't locate the user and I didn't know that the user had left.	26
43	One participant can change the name of the space without others knowing. (m)	Name of a room or space can be changed without others knowing it, which can lead to problems finding the room later. (All members of a room could see the name of the room changing or be notified with a sound or message box of its change - both could become annoying though)	12

OS	Out of Scope	The invitation window - the 'message' & 'response' text boxes look identical, even though you can't edit the text in the 'message' box. (Provide a visual indicator to show that it is static.)	1
		The program itself can be rather slow, thus causing times when one is listening to audio while the window is redrawing itself.	1
		I don't see the point for using this - maybe I am too frustrated!	2
		This program is very hard to use - I don't know where to start!	2
		The text chat facility displays a slight lag at start-up. The first message sent takes several seconds to appear on others users' screens. Subsequent messages appear in real-time. (This is likely a bug in the software.)	4
		There doesn't seem to be any way to keep another user out of one particular toolspace once they've accepted your invitation. (Allow optionally private or assignable and modifiable toolspaces so that you can allow or disallow others with relative flexibility.)	5
		The undo functionality in drawing mode keeps track of steps including work done in different windows. So if something was done in window 2, and currently at window 1, when one does an undo, it stays in window 1 and the person would not see any changes. What has actually happened however is the undo took place in window 2 without indication. Thus no feedback and no feedthrough. (Design to have independent window undo functionality.)	6
		Clicking on a person to find out about them sucks - why not have it tell you where they are.	9
		Don't realize that you have to be in the same space to talk to someone.	9
		Got into new area and don't know how I did it.	9
		Came across 'chat' - not obvious what it is.	9
		Don't know how to write in the Outliner.	9
		In Notepad, don't know who is writing what.	9
		Not obvious how to add lines.	9
		Don't see how to save workspace - if I leave when no one else is there will everything disappear?	9
		When I tried the instant message i.e. sending a message to a user, the from line defaulted to the first user in the list. I had to choose my own name. (When I decided to send a message, it should default to me in the 'from' line.)	10
		Text chat non-instantaneous for initial message. Unaware of initial message being sent.	12
		It is difficult to leave a space to go to another space or exit a meeting to meet other people. The method found was "delete" space but this implies it is to be permanently removed, but really it is still available for others still in the room. (Rename method to "leave" space or show its functionality better with graphical representation.)	12
		Although the chat box contains a history of communication, if users enter a task in progress they may have difficulty getting up to speed on communications occurred without the chat box history. (Could keep a voice history of the work which is transferred to the new user for optional listening.)	14
		Cannot tell what I sound like to others. (Play back to me an echo of my speech, may be dimmed slightly while I talk.)	18
		The verbal communication richness does not match the clunky physical interaction. (They need to be equalized: the levels of differences make them frustrating.)	22

NP	Not a problem (false positive)	In 'Author' column of the Outline tool, it used the computer name and not my name. (Use participant's name, not the computer's)	1
		It is unclear why there is even the option of 'half-duplex' communication. If the system offers the more useful 'full-duplex' why wouldn't that be the default?	5
		Has the inherent limitations of any system of the sort for starting informal meetings.	5
		Can hear person talking but can't tell if you are on talk	7
		Neither the chat nor the talk allows for overhearing conversations on commentaries. The chat has to be followed actively and the talk forces a person to listen to everything.	7
		With having the audio, chat and notepad, sending messages - there is an overload of communication methods. You might using one while others are using something else.	9
		Don't know everyone that is using chat, gives total number of people, but unless you see them write something, don't know they are there.	9
		Instant messages, no way to reply directly from message window.	10
		I cannot get voice messages from my partner while I am talking.	10
		When multiple users are speaking at the same time, either they all get cut off by the newest speaker or all the users are melded together. This causes a great deal of confusion and is pretty annoying. Creates a communication barrier. (Perhaps one could hear the first person to speak 1st , the 2nd person, and so on, rather than combining all the voices together. Want to create the atmosphere of being around a table where not individuals are heard at the same time.)	14
		Voice chat only allows one person to be talking at once - manual turn taking is very tedious. (Allow many people to talk at same time)	20
		Someone else started a space and I have yet to figure out why.	22
		Communication is not duplex; the only way to interject is to use the chat tool.	22
		With voice, action and immediacy is assumed. With only one person able to talk at once, communication is much more difficult. (Either remove verbal or add full duplex.)	22
		The record adding stuff is almost the same as chat.	22

B.3 Locales Framework heuristics– GroupDraw

Table B.3 GroupDraw Problems reported using Locales Framework heuristics

Heuristic 1 – Provide locales			
1	Unable to locate individuals not logged on to GroupKit.	No way to find people unless already in the program. (Link with IM)	18
2	Information does not persist upon closing the conference.	Persistence of conference is somewhat difficult to get.	23
		Information not saved from session to the next.	25
3	GroupDraw does not link well with an effective communication channel.	Components not integrated into a tool - chat exists as a separate tool.	24
		There is no means to find out how to establish a voice channel with other participant(s).	17
4	The conference model for a locale does not work well.	Although an application can be counted as a locale it felt more like collaborating over an application rather than a social locale.	21
		The "conference as locale" model has problems - are multiple GroupDraws one locale? Is each workspace a locale?	23
Heuristic 2 – Provide awareness (mutuality) within locales			
5	Unable to distinguish telepointers.	Tough to tell who's telepointer is which.	18
6	List of participants is not persistent enough.	Hard to keep list of people in a locale. (Put up a persistent status indicator of who is around and what they are doing.)	21
7	Limited awareness beyond active conference.	Limited awareness information in GroupKit.	23
		No knowledge of other locales in general.	23
8	Feedthrough of actions and objects is poor.	In GroupDraw, greater action and object feedthrough required.	23
Heuristic 3 – Allow individual views			
9	Cannot construct individual views.	Individual views do not exist - can't see differences in sizes of each user's view port.	18
		No way to zoom in and zoom out to concentrate on your own area.	20
		Poor support for individual views - no scrollbars, no way to determine another's view	23
		No individual views - group workspace only.	24
		No real way to have individual views.	25
		Individual views not present.	26
10	Unable to maintain awareness of other conferences outside the active conference.	Provides an individual view of space but in that view it is hard to keep up in what is happening in global view outside the local view. (Provide info on what is happening outside of local view.)	21
Heuristic 4 – Allow people to manage and stay aware of their evolving interactions over time			
11	No indication of prior actions and contributors.	Cannot see who changed something unless I track their telepointer visually. (Change awareness cues.)	18
		Rejoining the GroupDraw application later shows no information about previous users contributions. (History record showing which users entered/modified/exited the workspace.)	20
		Very little support for interaction trajectories - no history, no records of who has been there or what has happened.	23
		No sense of history - no explicit support for planning.	24

11	Cont'd	No historic information available	26
12	No mechanisms for planning future interactions.	No sense of history - *no explicit support for planning*.	24
		No planning mechanism - can't plan out future interactions.	26
Heuristic 5 - Provide a way to organize and relate locales to one another (civic structures)			
13	Cannot organize and integrate multiple conferences together.	I don't see any way to do this.	21
		All tools exist as separate windows and are not smoothly integrated.	24
		Integration of tools - tools are part of separate windows. This makes tools difficult to use together.	26
14	Unable to distinguish between conferences with the same name.	Multiple conferences of some name/type - hard to know which was the one I was supposed to join.	18
15	No means to transfer information from conference to another.	Cannot transfer information from one to another.	25

NP	Not a problem (false positive)	No knowledge of other open conferences.	23

B.4 Locales Framework heuristics – Groove

Table B.4 Groove Problems reported using Locales Framework heuristics

Heuristic 1 – Provide locales			
1	Tool-centric view makes it awkward to use Groove as a locale.	Arrangement of sites as workspaces is awkward - feels like a bunch of tools and one has to move from tool to tool. This tool-centered view does not work well. (Rooms metaphor and/or space containing tools would be much more natural.)	17
		Doesn't do it very well - confusing.	21
		A locale seems to be a giant selection of tools and spaces. Overwhelming! Spaces can grow very quickly. (Somehow simplify the interface for this.)	22
2	Communication means are inadequate and unreliable for sustaining teamwork.	The talk and chat tool are inadequate. The talk tool does not work reliably on all machines, and the text tool can be 'hidden'.	17
		Chatting is very disjoint. How could you get someone to open their chat window.	22
3	Awkward inviting new people into a space.	The need to invite members into a workspace is very awkward - it requires one to take explicit action to stay aware of who is online and so on. Much of our conversation consisted of trying to get ourselves established in a workspace. Very heavyweight.	17
		Hard to populate and invite contacts.	19
4	Difficult to distinguish between similar locales.	Different places look so similar it is difficult to tell what locale they are.	23
Heuristic 2 – Provide awareness (mutuality) within locales			
5	Cannot tell where others are if not sharing the same space.	From the Groove overview, the 'My spaces' area does not show who is in it.	17
		Indication of users in same place (change dot beside name) but no indication of where they are if not in the same place.	19
		If you have several users in different 'places' you can not see exactly where they are.	19
		No indication of where users are when in different locales.	26
6	Difficult to determine who is present in each tool within the active space.	The little numbers that show how many people are in a tool are very easily missed - therefore you don't know when people move to different tools.	17
		Cannot tell at a glance where (e.g. which tab) people are - have to mouse-over tab to get list.	18
		Hard to tell that someone else (or who) is in another tool (despite tab numbers)	23
		Difficult to know which parts of the workspace are being viewed by others.	24
		Can't tell exactly who is in Outliner with you, only # of people. Know who's in the overall space but not in specific areas.	25
		Cannot tell which areas others are working in.	26
		Cannot tell where other users are when they are using the same space as you.	26
7	No awareness of actions in other tools within the shared space.	The 'locale" is the current set of tools, *yet it is difficult to tell what is happening* within a tool (e.g. no telepointers, course updates) or *in a tool other than the one currently open.*	17
		Cannot tell which tool in a workspace has anything in it unless one visits it. This is heavyweight.	17
8	No awareness of actions or gestures within a tool.	The 'locale" is the current set of tools, *yet it is difficult to tell what is happening within a tool (e.g. no telepointers, course updates)* or in a tool other than the one currently open.	17
		Could not track others.	21
		You can't tell what people are pointing at. It is hard to indicate where you are.	22
		No immediate editing. *Cannot tell when someone is changing something.* (Lock field being edited/show edits as occur).	22
		Almost no awareness support inside a tool.	23

9	Actions do not appear until completed.	Cannot tell who is adding items as they add them - only that item is being added.	18
		No immediate editing. Cannot tell when someone is changing something. (Lock field being edited/show edits as they occur.	22
		Difficult to know what actions others are performing until they have been completed.	24
		Can't see that someone is modifying an item until the changed version appears.	25
10	Green indicator too subtle to indicate presence in a space.	The 'green light' is a bit too subtle to identify if people are present or not. Not sure of its meaning. Also, difficult to tell if someone has come in. (A stronger, simpler awareness mechanism).	17
11	Cannot determine availability of others on-line.	You see users on-line but not their availability.	19
12	No indication that someone is composing a chat message.	Cannot tell when people are composing chat messages - leads to perception lengthy delays in feedback or response. (So and so is typing a message - 'status indicators' as per MSN Messenger.)	18
13	No feedback that other participants have heard audio message.	Cannot tell if person heard my audio message - may not have had headphones on or volume turned down. (Cache messages and embed in chat transcript.)	18
14	No indication of who is in charge of 'Navigate Together' option.	Navigate Together is very frustrating when turned on. It is hard to tell who if anyone is in charge.	22
		Could use telepointer - cannot see who was responsible for navigation change in 'Navigate Together.'	18
15	No feedback regarding how one participant sounds to another.	Cannot tell what I sound like to others in audio chat. (Echo back my voice chat to me; show microphone volume controls; show others speaker level volume.)	18

Heuristic 3 – Allow individual views			
16	Difficult to manage the view of the space.	The workspace view is awkward and does not quite work. The organizational aspects are hard to maintain, especially because each workspace has a 'hierarchy' of contained tools. Very hard to manage a coherent individual view. (A better visual representation of a workspace and its contents is required, so one can make sense of it and manage it.)	17
17	No private chat facility.	No one-on-one private chat area.	18
18	No way to organize Outliner items into a personal view.	The way I want to organize items in the Outliner must be the same as others.	18
		It is not possible to 'hide' things that are not interesting to me.	19
		Does not allow users to customize individual views.	24
19	Navigate together action causes uncommitted actions to be deleted.	Navigate together can cause entered text to be deleted if not yet committed.	18
20	Two individuals can manipulate the same object simultaneously.	Views are very individual, you can edit something then someone else deletes it then your editing disappears along with the item mid-edit. (Lock out fields being edited. Show edits)	22

Heuristic 4 – Allow people to manage and stay aware of their evolving interactions over time			
21	Unable to maintain up-to-the-minute awareness of actions occurring within a tool.	Cannot tell when text and items are being added until person has finished. (Provide richer interactivity with feedthrough).	18
		Cannot see who changed what or what was changed on Outliner when I wasn't looking. (Need change awareness features.)	18
		There is no up to the minute information! If you can edit something, no one can tell until you are done.	22
22	No support of previous actions or visitors in a tool or workspace.	Cannot tell without visual inspection if anything in a particular tool has changed since the last time one looked.	17
		No support for newcomers to view changes made.	19
		No support for users to keep track of their past work.	19
		Calendar has entries that I didn't create, but how do I know who did? Also when did they get entered?	20

22	Cont'd	No history support (other than chat scrolling.)	23
		No indication of who was in the space or what happened until then.	23
		No sense of history - no facilities for planning future interactions.	24
		No way to track changes - only see current version.	25
		No historic information available - cannot tell what sequence changes happened, no archived versions.	26
23	No ability to plan future interactions.	No sense of history - *no facilities for planning future interactions*.	24

Heuristic 5 - Provide a way to organize and relate locales to one another (civic structures)			
24	No way to organize tools in a workspace.	Also similar to the above, its not clear what tools in a workspace are in use or how they are organized.	17
25	No way to see who was working in a space unless you were invited in.	It is unclear if you can see spaces or have to be invited.	22
		As far as I could tell, the only way to join a group was to have someone invite you. There are no way to see who was meeting or what they were working on.	27
26	Cannot link multiple tools together.	Cannot hyperlink Outliner items to web site or conversation transcripts in chat to organizer items.	18
27	Cannot aggregate multiple spaces together.	For the same reasons as previously mentioned, it is hard to create a good civic structure - one is left with a 'mess' of workspaces. Its unclear what they are for, who they are for, how they relate to one another, etc.	17
		Hard to relate new locales as we create them. Not possible to link two spaces together.	19
		There is no way I can aggregate 'places' together when related.	19
28	Cannot link Groove to other collaborative applications.	Poor links to outside collaborative applications.	23

NP	Not a problem (false positive)	Cannot determine why others appear offline when they are 'Ready to Groove'.	18