UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

EVALUATING GROUPWARE USABILITY AT THE COGNITIVE LEVEL OF HUMAN ACTION

António Manuel Silva Ferreira

DOUTORAMENTO EM INFORMÁTICA
ESPECIALIDADE DE ENGENHARIA INFORMÁTICA

2010

UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

EVALUATING GROUPWARE USABILITY AT THE COGNITIVE LEVEL OF HUMAN ACTION

António Manuel Silva Ferreira

DOUTORAMENTO EM INFORMÁTICA
ESPECIALIDADE DE ENGENHARIA INFORMÁTICA

2010

Tese orientada pelo Prof. Doutor Pedro Alexandre de Mourão Antunes

Abstract

This dissertation explores the importance of the cognitive level of human action in the evaluation and improvement of groupware usability. This research is motivated by the problem that current methods focus on the rational and social levels of human action and yet an increasing number of users relies on computers to fulfil collaborative tasks dominated by perceptual, cognitive, and motor skill.

The first contribution of this research is a groupware interface model that leverages existing knowledge on cognitive-level behaviour with single-user interfaces by expanding its application to multi-user interfaces. To do this, I show that the key differences between users interacting with the computer and interacting with other users through the computer can be supported by specialised groupware information flows and input/output devices.

The second contribution of this dissertation is a pair of methods for predicting groupware usability at the cognitive level of human action. The first method applies to scenarios of collaboration occurring routinely in shared workspaces. The second aims at capturing the intertwined nature of mixed-focus collaboration, encompassing shared and private workspaces. I use the methods to evaluate and compare the usability of competing designs in four scenarios of collaboration. The methods do not require user testing or functioning prototypes, so they can be integrated into the iterative process of interactive systems design.

The third contribution of this research is the evaluation of an attentive electronic brainstorming tool, which implements a novel attentive device that adjusts the delivery of group awareness information according to users' natural task switching between doing individual work and attending to the group. I present results from a laboratory experiment, which indicate that

groups produced 9.6 % more ideas when compared to the immediate broadcast of ideas and provide evidence suggesting that the usability improvement was due to the mitigation of information overload.

Keywords: evaluation, groupware, usability, attention.

Resumo (Portuguese Abstract)

Esta dissertação explora a importância do nível cognitivo da actividade humana, no qual as tarefas demoram segundos a realizar e são tipicamente repetitivas, na avaliação e melhoria da usabilidade de sistemas de trabalho cooperativo suportado por computador, também designados por *groupware.*

Estes sistemas de computadores permitem que grupos de interesse, como amigos e colegas, possam partilhar e organizar actividades de forma flexível e económica, onde o tempo e a distância deixam de ser obstáculos à colaboração. Alguns exemplos de *groupware* incluem os mensageiros instantâneos, usados por centenas de milhões de pessoas no mundo inteiro, os jogos multi-utilizador, que já atingiram cerca de dezasseis milhões de jogadores, bem como uma gama cada vez mais alargada de aplicações de escritório que estão a ser disponibilizadas na Internet. Com base nesta evidência, uma assumpção desta dissertação é que os sistemas de *groupware* estão a ficar cada vez mais ubíquos.

O problema abordado nesta investigação é que os métodos actuais de avaliação da usabilidade de *groupware* omitem o nível cognitivo da actividade humana, e, no entanto, as nossas características psicológicas, como a percepção, cognição, e capacidade motora, dominam a execução de tarefas de colaboração rápidas, mas normalmente muito repetitivas.

Uma consequência desta situação é que faltam instrumentos aos designers e investigadores de *groupware* que lhes permitam fazer optimizações de usabilidade de granularidade fina. Isto acontece porque os métodos actuais de avaliação da usabilidade visam tarefas colaborativas de relativa longa duração (que demoram minutos, horas, ou mais, a completar) e, portanto, baseiam-se em abstracções para conter o grau de complexidade da avaliação. Desta forma, as optimizações tendem a abranger vários passos de colaboração de

granularidade fina de uma só vez, o que causa problemas porque a usabilidade de sistemas de *groupware*, como na maioria dos sistemas computacionais, está inerentemente ligada aos detalhes da interface com o utilizador. Estas optimizações, mesmo que de pequena expressão individual, podem acarretar um efeito multiplicador significativo dado o crescente número de utilizadores de *groupware*, especialmente na Internet.

Outra consequência do nível cognitivo da acção humana ser negligenciado das avaliações de usabilidade de *groupware* é que o design da interface com o utilizador pode estar indevidamente alinhado com as características psicológicas humanas, o que pode fazer com que as tarefas colaborativas exijam uma carga de trabalho que excede as nossas capacidades limitadas de processamento de informação. Aliás, os utilizadores que realizam trabalho em grupo estão particularmente expostos a uma sobrecarga de informação porque têm de acompanhar o que se passa no grupo para além de realizarem trabalho individual, isto é, têm de dividir a atenção entre múltiplos fluxos de informação. Esta carga de trabalho pode penalizar a usabilidade dos sistemas de *groupware* devido ao aumento da probabilidade dos utilizadores não serem capazes de colaborar adequadamente.

Dada esta situação, a minha questão de investigação é: *como fazer avaliações ao nível cognitivo da actividade humana para melhorar a usabilidade de tarefas colaborativas realizadas através de sistemas de groupware?*

As avaliações de usabilidade ao nível cognitivo são bastante conhecidas no contexto das aplicações mono-utilizador, ao ponto de um conjunto de conhecimentos da psicologia aplicada ter sido reunido em modelos de engenharia de desempenho humano que predizem tempos de execução numa gama variada de tarefas de interacção pessoa-máquina. Estes modelos foram já, inclusivamente, aplicados no contexto de trabalho de grupo, mas sempre com a limitação de os utilizadores estarem restringidos a papéis individualistas e de a colaboração ficar de fora dos limites do sistema ou então ser abstraída. Em contraste, nesta dissertação estou interessado em avaliar as tarefas de colaboração realizadas através do sistema de *groupware.*

A primeira contribuição desta investigação é um modelo da interface do *groupware*, o qual alavanca o conhecimento existente sobre o comportamento humano com interfaces mono-utilizador, baseado em modelos de engenharia que predizem o desempenho humano, através da expansão da

sua aplicação a interfaces multi-utilizador. Para fazer isto mostro que as diferenças fundamentais entre os utilizadores interagirem com o computador (para trabalharem individualmente) e interagirem com outros utilizadores através do computador (para colaborar) podem ser suportadas por fluxos de informação e dispositivos de input/output especializados. Este modelo tem como propósito ajudar o designer a organizar o espaço de soluções numa gama alargada de sistemas de *groupware.*

A segunda contribuição desta dissertação é um par de métodos para avaliar a usabilidade de sistemas de *groupware* ao nível cognitivo da actividade humana. O primeiro método é aplicável a cenários críticos de colaboração que ocorram rotineiramente em espaços de trabalho partilhados e define usabilidade em termos do tempo necessário para executar tarefas colaborativas, tal como estimado pelos modelos de engenharia de desempenho humano. Na dissertação aplico este método para avaliar e comparar a usabilidade de alternativas de design em três casos de colaboração em espaços partilhados.

O segundo método visa capturar a natureza complexa e entrecruzada da colaboração que abrange tanto espaços partilhados como privados, bem como capturar os objectivos frequentemente conflituosos dos utilizadores enquanto estão a trabalhar individualmente ou quando estão a interagir com o grupo. Para fazer isto, combino estimativas de tempos de execução de tarefas com contribuições dessas tarefas para a progressão do grupo em direcção a um objectivo comum, em termos de produtividade individual, oportunidades criadas para os outros, e restrições para o trabalho de outros utilizadores. Na dissertação aplico este método a um jogo colaborativo, e mostro que, se para alguma outra coisa mais, este método serve para forçar o designer de *groupware* a pensar sobre as contrapartidas entre uma interface que permite aos utilizadores enquanto indivíduos serem mais produtivos e outra que permite um melhor desempenho do grupo enquanto um todo.

Os dois métodos de avaliação não requerem testes com utilizadores ou a construção de protótipos de *groupware* para produzirem resultados de usabilidade, o que atesta a sua natureza formativa, e permite a sua integração no processo iterativo de design de sistemas interactivos.

A terceira contribuição desta investigação é a avaliação da usabilidade de um sistema de *groupware* atentivo, que implementa um novo dispositivo de gestão da atenção humana, chamado *opportunity seeker* , o qual tem como

propósito mitigar a sobrecarga de informação em cenários de colaboração síncrona, isto é, em que todos os elementos do grupo estão a trabalhar em simultâneo. O *opportunity seeker* intercepta e guarda numa memória tampão a informação de estado sobre o grupo e ajusta automaticamente a entrega dessa informação a cada utilizador em função da alternância natural entre este estar a realizar trabalho individual e estar a prestar atenção ao grupo. Na dissertação mostro como este dispositivo pode ser adaptado e instalado numa ferramenta electrónica para geração de ideias, chamada ABTool, e como a fronteira entre os dois estados de atenção pode ser detectada através de actividade no teclado.

Para avaliar os efeitos do *opportunity seeker* na usabilidade da ferramenta ABTool, realizei uma experiência de laboratório em que pedi a grupos de voluntários para submeterem ideias em paralelo o mais rapidamente possível, e recolhi evidência de que quando os grupos estiveram sob a influência do *opportunity seeker* o número de ideias geradas aumentou em 9.6 % em comparação com a condição em que todas as ideias eram imediatamente difundidas por todos os utilizadores.

Adicionalmente, levei a cabo uma análise *post-hoc* que mostra que o *opportunity seeker* reduziu o número de entregas de ideias em 44.1 %, pois combinou as ideias em pequenos lotes, e que isso se traduziu em 54.7 % mais tempo para os utilizadores escreverem ideias sem serem interrompidos pela recepção de ideias de outros utilizadores. Nestas condições, os utilizadores foram 18.8 % mais rápidos a alternar entre a escrita de uma ideia, o que fizeram em 16.3 % menos tempo, e ler novas ideias de outros utilizadores.

Estes resultados evidenciam que o *opportunity seeker* criou condições para mitigar a sobrecarga de informação e mostram que a usabilidade de sistemas de *groupware* pode ser melhorada através de avaliações focadas nas limitações da capacidade de processamento de informação humana.

Com este conjunto de contribuições, mostrei que o nível cognitivo da actividade humana tem um papel determinante na avaliação da usabilidade de sistemas de *groupware*, complementando os níveis racional e social que têm sido tradicionalmente considerados por outros métodos de avaliação.

Palavras-chave: avaliação, trabalho cooperativo suportado por computador, usabilidade, atenção.

Acknowledgements

I would like to give thanks to my adviser, Professor Pedro Antunes, for his interest and availability to have discussion meetings, his continuous insistence on the writing and submission of papers, and his initiative to submit project proposals that financed travel and conference expenses.

Thank you also to Professor José Pino from Universidad de Chile for his contributions in two papers and his perseverance as corresponding author.

I also thank Valeria Herskovic from Universidad de Chile for helping with the staging and execution of the laboratory experiment, assisting on the data analysis, co-writing one paper, and discussing ideas.

My thanks also go to Bruno Coelho and Pedro Custódio for their logistics support, and to all volunteers who participated in the laboratory experiment.

Part of this research was supported by the Portuguese Foundation for Science and Technology, through project PTDC/EIA/67589/2006 and the Multiannual Funding Programme.

Dedicado à Sofia,
família, e amigos

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This dissertation describes a journey into groupware usability evaluation at the cognitive level of human action, where tasks relevant to collaboration take seconds to complete and are usually very repetitive, and aims at providing novel instruments of action and insights to designers and researchers.

This chapter situates the context of the dissertation and explains the importance of advancing the state of the art in groupware usability evaluation. I then frame the problem statement, formulate the research question, identify the objectives of this work, and describe the methods employed in this research. Finally, I outline the contents of the dissertation.

1.1 Context and Motivation

Groupware allows interest groups, such as friends and coworkers, to share information and organise activities in flexible and economic ways, where time and distance are no longer impediments to collaboration. This is made possible by a combination of three elements:

1. Communications networks, most notably the Internet, mobile data networks, local area networks in rooms and buildings, and others;
2. Personal computers, including desktop computers, laptops, personal digital assistants, and other variants; and
3. Software designed to be run collaboratively by multiple users, in the same or a different place, and at the same or different time.

The word *groupware* is, thus, connected to hardware and software for supporting group work. In contrast, *singleware* is a combination of hardware and software to assist the work done by a single user.

In this dissertation, I make the assumption that the *ubiquity* of groupware systems is bound to keep increasing because people and organisations are being more exposed than ever to a wider variety of applications designed for groups, due to the broad acceptance of tools, such as:

- The instant messenger, used by hundredths of millions of people world wide over the Internet (Leskovec and Horvitz, 2008);
- On-line multi-player games, which have reached sixteen million users all over the world (Woodcock, 2008);
- Electronic meeting systems, allowing groups to brainstorm, discuss, and vote ideas (Nunamaker et al., 1991);
- Free collaborative office applications available on the Internet, for instance, Google Docs and Zoho, which allow users to simultaneously work on the same documents;[1]
- Shared workspaces built on top of popular office software suites, such as Microsoft Office (Dragan, 2003), which enable users to share documents and collaborate in a familiar environment;
- On-line versions of those shared workspaces, which allow users to collaborate without installing software on the computer (Hackman, 2008);
- The Social Web, which allows users to collaboratively create an encyclopedia or lets people find information more efficiently by using voting mechanisms (Chi, 2008); and
- Web-based conference management systems (a commonplace for academicians), which support a variety of collaborative tasks, including the evaluation, discussion, and selection of papers.

The second and central assumption I make in this dissertation is that the importance of evaluating groupware systems at the *cognitive* level of human action is increasing relative to evaluation methods grounded on the rational and social levels (see Figure 1.1).

1 Google Docs is available at `http://docs.google.com` and Zoho can be tried out at `http:/www.zoho.com`, retrieved November 2008.

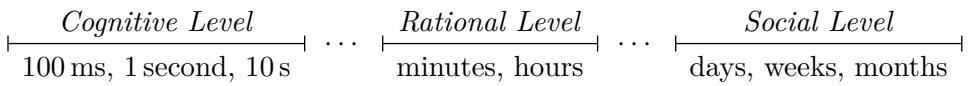


Figure 1.1: Levels of human action, adapted from Newell (1994, Fig. 3-3). Tasks at the cognitive level are quick and usually highly repetitive, such as pressing keys. At the rational level, tasks take longer to execute and are less repetitive, for example, on-line chatting. At the social level, tasks may take months and may not be repeatable at all, say coordinating a complex business deal.

The cognitive level of human action comprehends quick (from tenths of a second to several seconds) and usually highly repetitive tasks, in which human psychological characteristics, such as perception, cognition, and motor skill, play a significant role in task performance (Newell, 1994, Ch. 3). For instance, the succession of visual searches, key presses, and mouse cursor movements that might be needed to take ownership of a shared resource may largely dictate the time to complete this task.

In contrast, the other two levels of human action comprise tasks that are comparatively less frequent, and perhaps not repeatable at all, taking minutes to hours to complete at the rational level, or even days, weeks, or months at the social level. An example of the former is chatting on the Internet, and the latter may be coordinating a complex business deal.

By combining the two previous assumptions, I am arguing that:

> *It is increasingly important to evaluate groupware support for quick and repetitive collaborative tasks that are being performed by an expanding body of users and that are largely driven by our cognitive characteristics, because even small improvements to the execution of these tasks can have large net effects.*

It may seem paradoxical that in a dissertation about groupware evaluation the focus is neither on social and cultural aspects, nor on the workplace, not even on the organisation. After all, these contextual factors have long been proclaimed central to group performance (Grudin, 1988) and there is plenty of evidence in the literature to attest their popularity:

- Existing methods, namely evaluative ethnography, cooperation scenarios, and perceived value (which I describe in the related work) ground the evaluation on social and organisational factors;

- A review of groupware evaluations reveals that about half were field-based (Pinelle and Gutwin, 2000); and
- Some lessons from field research highlight the importance of the workplace in group performance, for instance, room lighting and seating configuration (Nunamaker et al., 1996).

However, the increasing ubiquity of groupware, especially on the Web, makes it more difficult to obtain generalisable results from field-based methods for several reasons, including:

- The user community may be very large and heterogeneous;
- The physical workplace in the organisation's premises may be replaced by any place with Internet access; and
- The group may be highly volatile, particularly when collaborative work is carried out by volunteers (as in the Social Web).

Moreover, field-based methods have been criticised for being less than appropriate for producing design solutions (Neale et al., 2004; Plowman et al., 1995) and, because they usually require experts to uncover work done through a functioning groupware system running in the actual workplace, they do not integrate well in the iterative design and prototype process inherent to interactive system design (Dix et al., 2003, Ch. 6).

Of course, some groupware evaluation methods are formative, that is, can be applied during the development or improvement of the system for the benefit of in-house staff (Scriven, 1981, p. 63). Examples include groupware heuristic evaluation (Baker et al., 2002) and walkthroughs (Pinelle and Gutwin, 2002), which share a conceptual framework called the 'mechanics of collaboration' (Gutwin and Greenberg, 2000). These mechanics describe group work primitives, such as write a message, overhear a conversation, and handover a resource to another user, which are then incorporated into the specific evaluation procedure of each method.

These methods can produce results within the time frame of a design and prototype iteration because the social and organisational aspects are ruled out of the evaluation. The argument for this approach is that problems in the human interaction with the groupware, for example, poor or nonexistent support for the mechanics of collaboration, 'will almost certainly guarantee its demise' (Gutwin and Greenberg, 2000), *irrespective* of the work context.

In this dissertation, I follow the same approach of grounding the evaluation of groupware systems on the *usability*, or ease of use, of the user interface. However, my focus is on the cognitive level of human action, rather than on the rational level, as currently happens.

In fact, existing formative groupware evaluation methods assume that the mechanics of collaboration are the finest level of activity, meaning they abstract *how* collaboration is actually executed by the users through the groupware interface. There are multiple ways to handover a resource, for instance, with different implications in user and group performance.

Interestingly, there is research on cognitive-level evaluation applied to group work, namely the groups of models approach (Kieras and Santoro, 2004) and distributed GOMS (Min et al., 1999), which are both grounded on the Goals, Operators, Methods, and Selection rules (GOMS) engineering model of human performance (Card et al., 1983). However, so far there has been an underlying assumption that group tasks are sequential and that each user is presumed to play an *individualistic* role.

Moreover, in both cases the act of collaborating (as in the mechanics of collaboration, mentioned earlier) was *not* evaluated at the cognitive level, but either expected to be carried out outside the limits of the computer system or replaced by an abstract communication operator derived from informal observations. In contrast, the focus of my dissertation is precisely on collaboration tasks done *through* the groupware system.

To conclude this motivation, I would like to add that some of the concepts presented in this dissertation inspired the proposal of project PTDC/EIA/67589/2006, which was accepted by the Portuguese Foundation for Science and Technology in August 2007 and is scheduled to end in mid 2010.

1.2 Problem Statement and Research Question

The problem addressed by this research is that current groupware usability evaluation methods overlook the cognitive level of human action, and yet our psychological characteristics, such as perception, cognition, and motor skill, dominate the execution of quick (up to several seconds), but usually highly repetitive, collaboration tasks.

One consequence of this situation is that groupware designers and researchers are lacking the instruments to make fine-grained usability optimisations. This is because existing groupware evaluation methods cover relatively long-running collaboration tasks (taking minutes to hours, or more, to complete) and so they rely on abstractions to keep the evaluation manageable.

Therefore, current optimisations comprehend multiple fine-grained collaboration steps at once, which have to be disambiguated and instantiated during groupware development, though with little guidance. This may cause problems because the usability of groupware systems, as in other interactive systems, is inherently linked to the *details* of the user interface.

Another consequence of the cognitive level of human action being left out of groupware usability evaluations is that the design of the user interface may be misaligned with our psychological characteristics, which may cause tasks to demand workloads that exceed our limited information processing capabilities, as illustrated in Eppler and Mengis (2004).

Actually, users doing group work are particularly exposed to *information overload* since they have communicate often to coordinate themselves with the others, have to deal with multiple information sources, and have to explicitly manage the alternation between doing individual work and keeping up with the group, including handling interruptions from colleagues. This extra workload increases the likelihood of users not being able to collaborate adequately and, thus, may penalise the usability of groupware systems.

From this situation, my research question is:

> *How to evaluate and improve groupware usability at the cognitive level of human action?*

This is question worth researching as even small usability improvements achieved at the cognitive-level of collaborative task execution can have large net benefits, given the increasing body of groupware users.

1.3 Objectives and Research Methods

The vision of this research is to contribute to better groupware usability through the practical applicability of cognitive-level evaluations. This vision will be supported by three research activities and corresponding objectives.

Objective 1: I will show that the cognitive level of human action can be useful to organise the design space of groupware interfaces.

This objective will be met by creating a model of the groupware interface that unifies cognitive-level information processing (perception, cognition, and motor skill) with computer support for collaborative interactions. This objective will be successful if the model can be applied in a range of groupware systems and if it can be integrated in usability evaluations.

Objective 2: I will show that cognitive-level evaluations can predict the usability of groupware systems in collaboration scenarios.

I will meet this second objective by constructing two usability evaluation methods based upon the following two common elements:

- Critical scenarios of collaboration, occurring frequently and affecting individual and group performance; and
- Existing engineering models of human performance, to characterise the collaboration scenarios at the cognitive level and provide quantitative performance estimates.

The methods differ in the way usability is defined and measured and in the type of collaboration that is supported, but in both cases the aim is to provide a systematic means for quickly comparing competing groupware designs, without requiring users or functioning prototypes. Thus, this objective will be achieved if the methods can contribute to formative groupware evaluation in a variety of collaboration scenarios.

Objective 3: I will show that focusing the evaluation on human information processing limitations can improve groupware usability.

This objective will be met in two steps: firstly, by designing and implementing a groupware device that adjusts collaborative information flows according to each user's state of attention to mitigate information overload during group work; and secondly, by conducting a laboratory experiment that compares groupware usability with and without the device in a fast-paced collaborative task. This objective will be successful if the experiment is valid and produces statistically significant results.

1.4 Overview of the Dissertation

This dissertation has nine chapters. After having identified the context, problem statement, and research objectives, in Chapter 2 I provide a review of evaluation types and explain how they have been applied to groupware systems. I also describe groupware evaluation methods and argue that they are appropriate for the social and rational levels of human action, except for laboratory experiments, which can cover cognitive-level tasks but are expensive to plan and execute, and thus less than suitable for quickly comparing design alternatives.

In Chapter 3 I review existing cognitive-level evaluation methods, and the underlying engineering models of human performance, which have long been used for evaluating the usability of single-user interfaces. I also discuss practical applications and limitations of this approach, with emphasis on previous evaluations in group work settings, which actually failed to consider collaborative tasks and, instead, were focused on independent tasks executed by users playing individualistic roles.

In Chapter 4 I propose a model of the groupware interface that leverages the insights about human behaviour provided by cognitive-level evaluation. I refer to the elements in this model all through the dissertation.

In Chapters 5 and 6 I present two evaluation methods grounded on engineering models of human performance and show how they can be applied to predict the usability of groupware interfaces in critical scenarios of collaboration. The first method focuses on collaborative tasks performed in shared workspaces, for which I provide three evaluation examples. The second method aims at capturing the intertwined nature of mixed-focus collaboration, encompassing shared and private workspaces, as well as the conflicting goals of users working as individuals or as elements of a group. I suggest three new usability dimensions, and show how to take measurements in an example application. In these two chapters I also discuss the merits and limitations of the methods, such as the assumption that users are tireless.

In Chapter 7 I expand the scope of the dissertation and highlight the need to evaluate groupware usability regarding human information processing limitations, particularly during information overload, a condition in which human attention starts discarding relevant information. I review related

work, including the goals and limitations of human attention, and existing attentive user interfaces, and conclude that most research does not address tackling information overload in group work settings.

So, in Chapter 8 I evaluate the usability of a custom-built groupware system that features an attentive device designed to mitigate information overload, and report on a laboratory experiment with groups of volunteers doing electronic brainstorming sessions, which shows that groups produced more ideas when they were exposed to the attentive device.

In Chapter 9 I look back at the research, summarise the main findings and contributions, and assess the fulfilment of the initial research objectives. I also synthesise some lessons I learned during the investigation, and suggest a number of directions for future work.

Finally, I also include two appendices with additional support materials: Appendix A explains the details of an equation that I used to exemplify one of the proposed groupware evaluation methods; and Appendix B contains samples of the materials used in the laboratory experiment, namely the consent form, the entrance questionnaire, and the instructions about brainstorming and the groupware tool.

Chapter 2

Groupware Evaluation

In this chapter, I present a review of the literature concerning groupware evaluation, organised in two parts: firstly, I describe *types of evaluation* and their purposes and show examples of how each type has been applied to groupware; and secondly, I survey *evaluation methods* adapted to or specifically created for groupware systems, and frame them according to the level of human activity that they cover.

2.1 Types of Evaluation

According to an evaluation thesaurus, evaluation is 'the process of determining the merit or worth or value of something; or the product of that process' (Scriven, 1981, p. 53). In other words, evaluation can be seen as a series of activities that ultimately produce an assessment report about services, products, work processes, or, of particular interest here, computer support for collaborative work—for the benefit of a target audience.

2.1.1 Formative Evaluation

When the target audience is the in-house staff and the evaluation is conducted during the development of a programme or product, it is called *formative* evaluation, and typically aims at improving something (Scriven, 1981, p. 63).

Some examples of formative evaluations applied to groupware systems are described in Paul and Morris (2009), Yankelovich et al. (2004), and

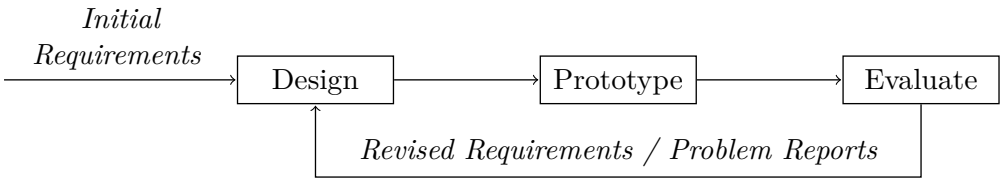


Figure 2.1: Formative evaluation in the iterative system design process, adapted from Dix et al. (2003, Figs. 6.5–6.7). In each iteration a new or revised design is prototyped and evaluated. The results of the evaluation guide the design of the next prototype. Mature prototypes become end-user systems, which can still be evaluated in a formative manner to identify improvements for future prototypes.

Prante et al. (2002). These three papers begin with a study of the problems with existing tools (through interviews with users, for instance), after which a set of requirements is specified or revised—for collaborative web searches, distributed meetings, and electronic brainstorming, respectively—which then guides the design and development of improved groupware tools.

Actually, this application of formative evaluation to groupware is very common because it matches the prevailing iterative system design, prototype, and evaluate process (see Figure 2.1), which is used in software engineering to overcome incomplete specifications of the users' needs at the beginning of a project as well as to diagnose and correct problems that arise during operation and maintenance of the system (Dix et al., 2003, Sect. 6.4).

More illustratively, according to a survey of papers published between 1990 and 1998, about 56 % of groupware evaluations were formative, mostly involving prototypes (Pinelle and Gutwin, 2000). Additional papers where this type of evaluation was applied to groupware can be found in Table 2.1.

One characteristic of all studies in Table 2.1 is that they describe multiple prototypes, whereas many other papers in the literature present and evaluate a single prototype and mention additional iterations in the future work, as, for instance, in Du et al. (2009) and Toups et al. (2009).

To conclude this section on formative groupware evaluation, I highlight the paper by Gruen et al. (2004), shown in Table 2.1, because it systematically reports on features/design, implementation/prototyping, and user study/evaluation of three e-mail client prototypes, including the links between the user feedback obtained from the evaluations and the new features of the subsequent prototypes, spanning a period of three years.

Table 2.1: Formative groupware evaluations, ordered by year. Each of these studies describes the evolution of a groupware system (or a particular feature) through various design iterations, guided by the evaluation of prototypes.

Study	Groupware Purpose	Design Iterations
Piper and Hollan (2008)	Communication facilitator	2
Szymanski et al. (2008)	Group guidebooks	2
Vassileva and Sun (2008)	Social file sharing	3
Piper et al. (2006)	Multi-player gaming	3
Tang et al. (2006)	Tabletop collaboration	2
Gruen et al. (2004)	E-mail communication	3
Langton et al. (2004)	Group programming	2
Herbsleb et al. (2002)	Instant messaging	2
Boehm et al. (2001)	Requirements negotiation	4
Nodder et al. (1999)	Instant messaging	3
Prinz and Kolvenbach (1996)	Workflow management	2
Tollmar et al. (1996)	Social availability	3
Ichikawa et al. (1995)	Video conferencing	2
Ishii et al. (1993)	Group writing	2
Baecker et al. (1993)	Group writing	2
Cool et al. (1992)	Video conferencing	3

2.1.2 Summative Evaluation

A different type of evaluation applies when the purpose is to decide about something, mostly after the completion of a programme or product and usually for the benefit of some external audience or decision-maker. This is called *summative* evaluation (Scriven, 1981, p. 150), which, referring again to the review by Pinelle and Gutwin (2000), represented about 38 % of all groupware evaluations for the surveyed period.

Table 2.2 synthesises the main differences between the formative and summative types of evaluation, including when they are generally applied, their chief purpose and focus, and their typical target audience.

I note that summative evaluation can also be conducted in-house and during the development of a product or programme, such as when a manager verifies the progress of an ongoing project against a set of predefined design goals. This is not formative because there is no intention to make improvements but rather to produce a status report that can be used to decide about the future of the project or product, as in Kammer et al. (2000), Cockburn and Dale (1997), and Sohlenkamp and Chwelos (1994).

Table 2.2: Formative and summative types of evaluation: 'when the cook tastes the soup, that is formative; when the guests taste the soup, that is summative.' (Scriven, 1981, pp. 63 and 150).

Evaluation Type	When Applied	Purpose	Focus On	Target Audience
Formative	During development	Improve	Process	In-house
Summative	After completion	Decide	Outcome	External*

*May also be conducted for the in-house staff, for example in the form of project status reports.

The next examples of summative groupware evaluations take place in the more characteristic case in which a decision must be made about the adoption or rejection of a prospective groupware tool. Perhaps unsurprisingly, this decision may turn out to be difficult to justify for several reasons:

- The tool under trial is adequate for recurrent tasks, but is less successful in occasional complex tasks (Bossen, 2006; Tammaro et al., 1997);
- The groupware is regarded as an organisational success, but also as a technical failure, or the other way around (Blythin et al., 1997);
- Some users benefit from using the tool whereas others feel their work has become more difficult (Grudin, 1994; Rogers, 1994); and
- Users freely adopt the tool during a trail period, even though they cannot explain why (Bjerknes and Bratteteig, 1988).

Grudin (1994) identifies additional factors that complicate groupware adoption decisions, namely the difficulty in learning from past decisions, which restricts the use of intuition, and the uncertainty about the rate of adoption, necessary for the tool to be useful to the entire organisation.

In contrast, the rejection of groupware can be summary, as reported in Parent and Gallupe (2001): all it took was the feeling that the proposed group support system was best suited to groups undergoing internal conflicts—an image the decision-maker did not want to pass to others since his people were working well together.

Returning to groupware adoption, despite the difficulties mentioned earlier, a typical path taken by decision-makers is to gather a set of requirements (sometimes called critical success factors), then search the market for systems that match those requirements, and finally ask users if they would prefer switching to a new system. This path is described in Mosier and

Tammaro (1997) and Tammaro et al. (1997), regarding the adoption process of electronic meeting schedulers and collaborative writing tools.

In fact, the authors of those two companion papers carried out a more complete summative evaluation by gathering lists of perceived benefits of the groupware tools under trial, such as being easier to know people availability or the production of higher quality documents, respectively. Another list of benefits can be found in Niederman and Bryson (1998), which also covers the costs introduced by a meeting support system, as perceived by its users.

The last example of a summative evaluation concerns a decision about the continuation or decommissioning of an active groupware system based upon its usage over time. The general idea is that if the user base is steady or growing, then the groupware (or some of its features) is still being useful and should continue to be supported. This eventually happened with a system for broadcasting instant messages in a multinational organisation, when its user community fervently reacted against the shutdown of the system, claiming it fulfilled critical business needs (Weisz, 2006).

2.1.3 Comparative Evaluation

One of the most popular types of groupware evaluation is the *comparative* type, which answers the question (Damianos et al., 1999): is system A better than system B? This situation is justified by the confluence of several factors, including the following:

- Comparisons are compatible with both formative and summative evaluations, which means they can be done between several prototypes during the various iterations of the software development process or between functioning systems competing for adoption or for continuing in active service;

- A standard way of improving the design of systems is to test the effects of some candidate features on behaviour and check the direction pointed by the relative differences in performance (Sanders and McCormick, 1992, pp. 24–25). Moreover, these differences may trigger a search for explanations and lead to more feature comparisons in order to choose the preferred explanation (Abelson, 1995, pp. 3–6);

- Product evaluations, as those found in consumer reports or those performed in acquisition programs, compare the merits of multiple critical competitors, so that the best value for money (or other resources) can be ascertained (Scriven, 1981, pp. 39–40); and
- Comparative evaluations are also used to highlight the advantages of some proposed system or feature relative to the state of the art. In fact, this is a customary form of substantiating a contribution to the body of knowledge in research.

A characteristic of all comparisons is the need to set up *common criteria* upon which to base the determination of the value or worth of the options. Naturally, the selection of these criteria depends upon the types of questions the evaluation must answer and ultimately involves defining either a set of dimensions upon which to grade the systems, or, particularly in experimental research, defining which specific, operational, measures should be collected (Scholtz and Steves, 2004).

Comparative Evaluations in Experimental Research

One comprehensive review of the measures used in groupware experimental research lists 120 distinct dependent variables,[1] including decision time, number of decision cycles, system utilisation, number of errors, number of comments, learning time, and many others (Fjermestad and Hiltz, 1999).

Such a variety of measures has lead to the creation of *groupware evaluation frameworks* that consider high level, conceptual, measures, and arrange them in categories. So, for instance, number of comments is abstracted into effectiveness, and this, in turn, belongs to the task outcome measures category. The following is a brief overview of some of the existing frameworks:

- Araujo et al. (2004) suggest measuring the usability of the groupware system, the level of collaboration that can be achieved, and the cultural impact of using the system;
- Antunes and Costa (2003) consider task, group, organisation, and technology categories, and their corresponding measures, such as efficiency, effectiveness, satisfaction, perceived value, and economic impact;

1 Dependent variables answer the question: what do I want to observe? They are the data, that is, the results of the experiment—what is measured (Howell, 2007, p. 4).

- Damianos et al. (1999) propose a framework with four layers that, at the top, aims at measuring how well the groupware supports work tasks, transition tasks (for instance, setting up a meeting), social protocols, and group characteristics (such as, size and space/time collocation);
- Tung and Turban (1998) place decision quality, completion on time, and other measures in the task category; cohesiveness, coordination competence, communication effectiveness, and more in the group category; and status, culture, and structure in organisational measures.

All these frameworks are designed to organise and bound the evaluation space by purposefully identifying only the relevant types of data that should be collected and compared, though, in practise, they are considerably extensive. From these circumstances, I agree with Sanders and McCormick (1992, p. 34) in that 'any attempt to classify criterion measures inevitably leads to confusion and overlap.'

In fact, most groupware experimental research does *not* make use of the full spectrum of measures proposed by existing frameworks and, instead, prefers to focus the evaluation on a relatively small number of conceptual measures, particularly task effectiveness/efficiency, user satisfaction, and process gains/losses, as synthesised in Fjermestad and Hiltz (1999).

This situation is corroborated by the literature survey in Table 2.3, covering almost twenty years of comparative groupware evaluations in experimental research, which reveals that the majority of the studies that compared at least three options (in most cases, generic groupware features rather than complete systems) report results for up to five operational measures.

The more elaborate studies in Table 2.3 consider between six and fifteen operational measures, and more than that in one case. When taken collectively, these measures can be arranged into the four categories proposed in Antunes and Costa (2003), though none of the studies comprised them all. For illustration purposes, I provide the following examples:

- Davey and Olson (1998) compared three decision support systems along measures from the *task* category, such as time to reach a solution and number of solutions considered in a bank investment decision;
- Bose and Paradice (1999) measured attitude towards group judgement-making process and perceived degree of consensus, among others, in a

Table 2.3: Comparative evaluations in groupware experimental research, ordered by year. These studies consider at least three competing options (for terseness sake) being compared along a number of operational measures—that actually produced data. I use number intervals to emphasise the point that most groupware experiments regard relatively few measures. Most studies compared generic groupware features and not complete systems, which is common in laboratory settings.

Study	Groupware Purpose	Options	Measures
Pinelle et al. (2009)	Tabletop collaboration	4	6–15
Tuddenham and Robinson (2009)	Tabletop collaboration	3	6–15
Balakrishnan et al. (2008)	Problem solving	4	6–15
Stuckel and Gutwin (2008)	Delay mitigation	3	1–5
Fraser et al. (2007)	Display trajectories	4	1–5
Nguyen and Canny (2007)	Video conferencing	3	1–5
Pawar et al. (2007)	Group learning	4	6–15
Söderholm et al. (2007)	Telemedicine	3	1–5
Forlines et al. (2006)	Group searching	3	1–5
Hauber et al. (2006)	Video conferencing	4	>15
Ranjan et al. (2006)	Remote assistance	3	1–5
Nacenta et al. (2005)	Tabletop collaboration	6	1–5
Tsandilas and Balakrishnan (2005)	Interference mitigation	4	1–5
Gutwin et al. (2004)	Delay mitigation	3	1–5
Fussell et al. (2003)	Remote physical tasks	5	6–15
Yuan et al. (2003)	Web-based negotiation	3	6–15
McNee et al. (2002)	Group filtering	6	1–5
Yang and Olson (2002)	Group navigation	4	1–5
Garau et al. (2001)	Avatar expressiveness	4	1–5
Zanella and Greenberg (2001)	Interference mitigation	3	1–5
Connell et al. (2001)	Impression management	3	1–5
Fussell et al. (2000)	Remote physical tasks	3	6–15
Mennecke et al. (2000)	Decision and negotiation	4	1–5
Bose and Paradice (1999)	Group decision	3	6–15
Davey and Olson (1998)	Group decision	3	6–15
Zhao and Stasko (1998)	Filtered video	5	1–5
Aytes (1995)	Group drawing	3	6–15
Hymes and Olson (1992)	Brainstorming	3	1–5
Gale (1991)	Shared whiteboard	3	6–15

comparison between two group decision support systems and the option of not using computers; these measures fit into the *group* category;

- Gale (1991) compared three variants of shared whiteboards and asked users to estimate the time savings and productivity increases if the department or whole site had installed each system, separately; thus, this evaluation used measures from the *organisation* category; and

- Hauber et al. (2006) measured the usability of three techniques for video conference systems through user questionnaires and compared it with the usability of face-to-face meetings. In this case, the measures were from the *technology* category.

I note that some comparative evaluations in groupware experimental research consider well over fifteen operational measures, but the comparison is frequently made between using or not using a groupware tool and, sometimes, between using or not using computers to collaborate. Thus, these studies tend to be less focused on technology and more interested in the other categories of measures, particularly task- and group-related.

The best example I could find of this disparity is the study by Michailidis and Rada (1994), which compared a single-user text editor with a group writing tool along *seven* usability measures (overall adequacy, breakdowns caused by the tools, and support for five aspects of coordination) and *sixty-seven* measures concerning the writing task, ranging from text length and complexity, document and content quality, style clarity, and more.

Comparative Evaluations Outside Experimental Research

In contrast with the previous cases, comparative groupware evaluations *outside* the domain of experimental research tend to be more focused on technology and how well the systems or tools help fulfil group tasks.

This often happens in discussions concerning the state of the art, for instance to demonstrate that no system supports a feature being proposed (Aneiros et al., 2003; Roussev et al., 2000; Munson and Dewan, 1994), to introduce improvements (Sun, 2000; Baecker et al., 1993), to automatise procedures (Wang et al., 2007), or to draw a map of the degree of technological development across a range of prototypes (Peng, 1993).

Other examples of groupware comparisons assess the level of support for task activities, for instance the possibility of facilitating pre-meeting arrangements (Antunes and Ho, 2001; Dubs and Hayne, 1992) or of organising ideas after brainstorming sessions (Nunamaker et al., 1991).

Finally, I only found one case in which groupware systems were compared along a list of design guidelines (Kruger et al., 2004), which is, perhaps, surprising given the abundance of recommendations that authors often feel

Table 2.4: Groupware comparisons outside experimental research, ordered by year. These studies compare tools along a set of dimensions, usually to highlight the advantages of a new tool being proposed or to assess technological support for a list of requirements.

Study	Groupware Purpose	Options	Dimensions
Wang et al. (2007)	Group awareness	7	1–5
Kruger et al. (2004)	Tabletop collaboration	6	6–15
Aneiros et al. (2003)	Group browsing	8	1–5
Antunes and Ho (2001)	Meeting preparation	6	1–5
Roussev et al. (2000)	Distributed infrastructures	8	6–15
Sun (2000)	Group undo	3	6–15
*Bose and Paradice (1999)	Group decision	3	6–15
Munson and Dewan (1994)	Object merging	6	6–15
Baecker et al. (1993)	Group writing	7	>15
Peng (1993)	Group drawing	13	>15
Dubs and Hayne (1992)	Group facilitation	7	6–15
Nunamaker et al. (1991)	Electronic meetings	13	1–5

*This study also compared the same groupware systems in an experiment (see Table 2.3).

tempted to make at the end of an evaluation. As Scriven (1981, pp. 131–132) puts it, 'a road-tester is not a mechanic,' and the complexities of computer-supported cooperative work may be a fine example of useful recommendations requiring 'not only local knowledge but very special skills.'

Table 2.4 shows that groupware comparisons made outside experimental research typically comprise a significant number of options (up to thirteen tools side-by-side) and that the assessments consider a relatively high number of dimensions, in contrast with the measure counts in Table 2.3.

Another difference is that the evaluations in Table 2.4 use the categorical scale of measurement, with simple values such as yes/no, low/medium/high, none/partial/full, or domain-specific names, whereas most of the comparative studies in Table 2.3 use interval or ratio scales, with numerical values.

2.1.4 Other Types of Evaluation

The evaluation thesaurus (Scriven, 1981) mentions several other types of evaluation that have been applied to groupware systems. I describe here two dichotomies, the first of which is the popular quantitative/qualitative division, and the second is between holistic and analytic evaluations.

Quantitative Versus Qualitative Evaluations

Quantitative evaluations are based upon numerical measures or numerical analysis of categorical or ordinal data (Scriven, 1981, p. 126). So, when a researcher measures task completion times, s/he is following the quantitative path, and the same happens when s/he summarises the results of questionnaires using counts or percentages of each possible answer. Almost all studies in Table 2.3 are of this type.

It is usual in quantitative evaluations that large amounts of data be processed, and treated without special consideration for the context where they were captured. In contrast, *qualitative* evaluations do not seek the statistical significance permitted by large bodies of data and, instead, aim at making long-term, in-depth, studies with small samples of people performing relevant collaborative tasks (Miles and Huberman, 1994, p. 27).

One reason why qualitative evaluations last weeks, months, and, sometimes, years, is that one of its main goals is to find out how well the interactive computer system fits with existing work practises. Thus, an essential part of the evaluation involves understanding the rhythms of work routines, the movements of people and artifacts, and the social nature of work, including how people coordinate their activities (Martin et al., 2005). Naturally, this can become very time-consuming and even unpredictable when the evaluator assumes a minimally intrusive role in the workplace being studied.

The literature describes several qualitative evaluations applied to groupware, typically conducted for large organisations and confined to a physical space. A recurring theme is the evaluation of time-critical systems in intense collaboration workplaces, such as hospital wards (Tang and Carpendale, 2008; Bossen, 2006; Martin et al., 2005; Heath and Luff, 1996), air traffic control rooms (Twidale et al., 1994; Harper et al., 1989), and telecommunications centres (Whittaker and Amento, 2003). The evaluations conducted in these, sometimes hectic, workplaces, usually show that when the groupware does not mesh well with existing work practises, it is easily rejected by the users, who quickly revert to using the old system, even if paper-based.

Curiously, some qualitative evaluations report that, even though problems were detected, the groupware system continued to be used because it facilitated accounting (Bowers et al., 1995; Rogers, 1994).

Finally, I note that some of the summative evaluations mentioned in Section 2.1.2 are simultaneously qualitative and that more pointers can be found in Pinelle and Gutwin (2000), which also reveals that about 72 % of all groupware evaluations surveyed were of this type, versus 7 % for quantitative, and the remaining 22 % for both types being used in the same study.

Holistic Versus Analytic Evaluations

Another dichotomy in evaluation, less popular, is between the holistic and analytic types. The former means doing an assessment at the macro-level, with no need to look into the details to obtain an overall value (Scriven, 1981, p. 72). The latter is done at the micro-level and assumes the value of the whole is made up of the evaluations of its parts (Scriven, 1981, p. 7).

It could be argued that some qualitative evaluations are *holistic*, especially those that are also summative, not the least because this certainly happened in two assessments, which I recall from Section 2.1.2: the first concluded users freely adopted a groupware system for no particular reason (Bjerknes and Bratteteig, 1988), thus making it difficult to ascertain the factors that contributed to success; and in the second, there was no need to evaluate the parts of a negotiation tool because the whole system was summarily rejected by a decision-maker based upon his feeling (Parent and Gallupe, 2001).

Conversely, *analytic* evaluations are at the core of many comparative assessments. For instance, the dimensions alluded in Table 2.4 are a way of spreading out the value of competing groupware systems into their constituent parts, to allow comparisons to be carried out at a more detailed level.

Similarly, the evaluation frameworks discussed on page 16 are another way of decomposing the value of a groupware system into multiple categories of measures, which, by design of the frameworks, are all essential to ascertain the overall system worth.

Furthermore, because analytic evaluation helps detect the individual components that drive the success or failure of a system, it may be a useful approach to formative evaluation (Scriven, 1981, p. 25). In fact, the iterative nature of interactive system design relies on the identification of problems in specific parts or features, which are subsequently addressed in the future prototypes (see Section 2.1.1).

2.2 Evaluation Methods

The types of evaluation in the previous section provide a conceptual overview of the variety of purposes, target audiences, placements in the software development process, outcomes, and levels of detail afforded by evaluations. In this section, I describe operational methods to systematically gather data and produce results in groupware evaluations.

Some groupware evaluation methods are adaptations of methods originally designed for the assessment of single-user (singleware) tools. This is natural, and will likely continue over the next years,[2] given that groupware evaluation can be regarded as a super-set of singleware evaluation. In addition, there is still a large gap between the number of methods in these two domains: over a *hundredth* for singleware evaluation (Ivory and Hearst, 2001) compared with less than *thirteen* for groupware (Herskovic et al., 2007).

Other methods were specifically created for, or gained notoriety with, the evaluation of groupware systems, mainly by considering the organisational and social contexts as prime components of the assessment, and, contrary to the focus of the methods I alluded in the previous paragraph, by dispensing with the classical human-computer interaction measurements, unless deemed necessary. I explore this duality in the next subsections.

2.2.1 Context-Based Methods

One of the most important trends in groupware evaluation is based upon the assumption that social, cultural, workplace, and organisational factors are determinants of group performance and groupware acceptance, so evaluations should be carried out through field studies in the 'real world' instead of laboratory experiments in artificial settings (Grudin, 1988).

As a consequence of this assumption, one commonly accepted purpose of evaluations, according to this trend, is to inform about the success and failure of functioning systems, that is, to check if the groupware fits with existing work practises (Martin et al., 2005) and delivers what is required of it in the real context of use (Hughes et al., 1994).

2 To reinforce this point, the methods I propose in Chapter 5 and 6 reuse an analytical method that was originally devised for evaluating single-user tools in the early 1980s.

The popularity of this trend is well documented in the literature. For example, the aforementioned paper by Grudin (1988) was the second most cited over the first twenty years of a major computer-supported cooperative work conference (Jacovi et al., 2006). Another evidence is that a review of groupware evaluations revealed that about half were field-based, about the same proportion as laboratory experiments (Pinelle and Gutwin, 2000).

Finally, some evaluation methods for groupware systems are grounded on contextual factors, which I have organised in evaluative ethnography, cooperation scenarios, and perceived value.

Evaluative Ethnography

Ethnography is concerned with producing detailed descriptive accounts of the everyday life of the people who are the subject of a study. It requires one or more ethnographers working in the field, who immerse themselves in the culture under investigation for prolonged periods of time (weeks, months, and even years) to make visible the real world interactions of a community.[3]

This contextually-rich characterisation inspired some of the first groupware evaluations, which applied ethnography in the workplace to confront how designers expected the system to be used versus how the users really collaborated (Hughes et al., 1994; Twidale et al., 1994).

Interestingly, those early evaluations were adaptations of formal ethnographic studies in that the duration of the studies was restricted to between two and four weeks and also in that the work of the evaluator was more informal and opportunistic, accounting for less details of the workplace. Another adaptation reported more recently is to conduct the evaluation in discrete phases, such as before and after system deployment (Tang and Carpendale, 2008), instead of the more usual continuous workplace immersion.

These simplifications make evaluative ethnography more manageable and less costly, but even so it is criticised for being expensive and unsuitable for rapid prototype evaluation (Gutwin and Greenberg, 2000). Another critique is that it has been considered less than appropriate for producing design solutions because the translation between the discursive language used in

3 From the definition of *ethnography* in Encyclopedia Britannica, retrieved December 2008, `http://www.britannica.com/EBchecked/topic/194292/ethnography`.

ethnography and the design blueprint may get distorted or misconstrued (Plowman et al., 1995), which, putting it another way, could simply mean that the 'implications for design' typically suggested at the end of ethnographic evaluations are not understood by the groupware audience (Dourish, 2006).

Also, its focus on the description of the present state of the workplace contrasts with the problem-solving, future-oriented, nature of design (Neale et al., 2004). And, finally, there is the problem of scale, which is particularly pertinent in the heterogeneous, and physically distributed, workplaces that are emerging as the computer becomes increasingly ubiquitous and portable (Crabtree et al., 2009), and even more so with the large and volatile groups that populate many of the collaborative efforts on the Web (Chi, 2008).

Cooperation Scenarios

Evaluation based upon cooperation scenarios couples field-based studies with user interviews to elicit new groupware features, which are ultimately validated in workshops with the users (Stiemerling and Cremers, 1998).

The purpose of the interviews in the workplace is to extract contextual information, especially work practises, motivation and goals for cooperation, roles played, and tasks performed. From this, cooperation scenarios are built and, if necessary, refined by interviewing more users.

Next, scenarios are examined for problems with the current work practises, possibly explicitly stated by the users themselves, and design solutions are created, that is, new cooperation scenarios are proposed.

A preliminary evaluation of these scenarios is then conducted based upon role-oriented analysis (without any users) to predict who would benefit the most and to estimate task workload for all involved parties.

Finally, the new cooperation scenarios are implemented in the groupware system and evaluated by the end users through a discussion workshop, where unexpected design flaws can be discovered.

A variant of cooperation scenarios adds claims analysis and frequency of use to the evaluation (Haynes et al., 2004). Claims are statements about the positive and negative effects of using the groupware system in a scenario, as stated by the users in interviews. In addition, the number of times a scenario (occurring in everyday work or envisioned) is mentioned in the interviews

is counted, which, together with claims analysis, are then used to establish priorities for improving the groupware system and to assess how far it is from the users' expectations.

Perceived Value

Another method that builds up from the users' opinions is called perceived value, which differs from the previous by shifting the focus of the evaluation away from planned scenarios of use and into open exploration of the groupware system (implicitly situated in the organisational and group context), as well as by involving another type of stakeholder, namely the software developers (Antunes and Costa, 2003).

The method begins with developers identifying system components that should be visible to the users and that are considered relevant to the evaluation, based upon a preliminary appreciation of the importance of the technology in the specific organisational context.

Next, a list of concrete evaluation attributes is negotiated between the developers and the users. Currently, a predefined list of attributes exists for meeting support systems, covering roles, processes, and resources of the system at the individual, group, and organisational levels.

Finally, users experience the groupware in a free manner, after which they fill out an evaluating matrix correlating the perceived contribution of the system components to the attributes. This can be done either individually or collectively, in a meeting.

2.2.2 Usability-Oriented Methods

The usability trend in groupware evaluation is based upon the assumption that if users have difficulties interacting with the groupware they will likely abandon it and switch to other alternatives that also get the work done, *irrespective* of the work context (Gutwin and Greenberg, 2000).

Thus, the focus of usability evaluation is on aspects such as the effectiveness, efficiency, and satisfaction achievable by users while performing collaborative tasks through the groupware system.

Usability evaluation is well-known in the Human-Computer Interaction (HCI) field and has been extensively applied to single-user interfaces. Some of

the most popular usability evaluation methods include conducting laboratory experiments, checking the interface against a set of heuristic guidelines, and finding problems via walkthroughs of the actions the user would have to perform to complete a task (Dix et al., 2003, Ch. 9).

These methods have been adapted to groupware usability evaluation for different reasons. Laboratory experiments enjoy almost universal acceptance in science, so the transition to the groupware domain is natural, but they are costly and difficult to setup. The other two methods purposefully reduce the complexity of the evaluation at the expense of precision and realism.

Laboratory Experiment

The literature on groupware usability evaluation includes numerous references to laboratory experiments, some of which shown in Table 2.3 on page 18. Moreover, Pinelle and Gutwin (2000) reveal that about half of all groupware evaluations surveyed were experiments and that approximately 70 % of those were assessments of the user interaction through the system.

With this method, the evaluator asks groups of volunteers to perform a collaborative task via a groupware system/prototype installed in a laboratory room. Each session is carefully prepared to minimise external influences and to keep the results precise and comparable.

At the end of the experiment, the evaluator analyses all recorded evidence to determine if there is a link between preselected usability measures and the computer features being tested, guided by the hypotheses that originally motivated the evaluation. Some common usability measures include the effectiveness, efficiency, and satisfaction achievable by the users while collaborating through the interface, which are typically assessed from empirical measurements and questionnaires.

Despite the popularity of laboratory experiments as a method for evaluating groupware usability, the fact is that it is particularly difficult and costly to perform with *groups*. This is due to several reasons:

- Groups may be hard to find with the required competencies, may be geographically distributed, or may simply be unavailable for the time necessary to accomplish the collaborative tasks, which precludes the experiment from being executed in a controlled laboratory room;

- A functioning groupware system or prototype is needed, as well as a large paraphernalia of software instruments to gather data; and
- The experiment itself requires significant time and expertise to setup and execute, including devising a protocol plan, preparing materials, arranging the laboratory room, and performing the sessions with users.

Curiously, besides the precision and comparability pursued by experimenters, which are indispensable for replicating the tests and increase the confidence in the results, Greenberg and Buxton (2008) allude that another factor contributing to the success of laboratory experiments could be the existence of a methodological bias in academic review committees, possibly at the expense of creativity and innovation.

* * *

In contrast with laboratory experiments and context-based methods, the next two groupware usability evaluation methods can be applied *without users* and *without functioning systems*. In fact, the motivating idea is that simple mock-ups, for example, made of paper, may be sufficient to detect most usability problems.

This means that these methods require less time and effort to conduct, and so they enable more iterations through the design, prototype, and evaluate process (see Figure 2.1 on page 12). Another way of putting this is that usability problems can be detected and repaired earlier.

In both methods, group work is represented using the same conceptual framework, called the 'mechanics of collaboration' (Gutwin and Greenberg, 2000), which I briefly describe before presenting the methods themselves.

The mechanics of collaboration are group work primitives, such as write a message to one or more users, overhear a conversation, know who is currently on the group, and handover a resource to another user (see more examples in Table 2.5). Their purpose is to describe general purpose communication and coordination activities that are needed for users to collaborate.

The mechanics are designed to provide a comprehensive view of group work, although they may be, and have been, revised and expanded (Pinelle and Gutwin, 2008; Pinelle et al., 2003). They are also purposefully observable, so that they can be evaluated one at a time, which might be useful in the breakdown of complex collaborative tasks.

Table 2.5: Mechanics of collaboration, adapted from Pinelle et al. (2003, Table 1).

Type	Category	Mechanic	Typical Actions
Communication	Explicit communication	Spoken messages	Conversation
		Written messages	Persistent conversation
		Gestural messages	Drawing, demonstrating
		Deictic references	Pointing and speaking
	Information gathering	Basic awareness	Attending to the group
		Feedthrough	Changes to objects
		Consequential com.	Body position, gaze direction
		Overhearing	Presence of talk
		Visual evidence	Confirmation of understanding
Coordination	Shared access	Obtain resource	Take objects, occupy space
		Reserve resource	Notify others of intention
		Protect work	Notify others of protection
	Transfer	Handover object	Give/take object
		Deposit	Place object and notify

Groupware Heuristic Evaluation

This method builds upon a list of usability heuristics created for single-user interfaces, such as 'provide feedback' and 'use the user's language' (Nielsen, 1992), by introducing new ones derived from the mechanics of collaboration (Baker et al., 2002, 2001). These complementary heuristics are guidelines believed to reflect good usability practises for human-computer interaction with groupware systems, such as 'facilitate finding collaborators,' 'provide protection,' and 'support people with the coordination of their actions' (compare with the mechanics of collaboration in Table 2.5).

The method begins with multiple experts visually inspecting the groupware interface (which may be almost anything available, from a functioning system to a paper mock-up) and judging its compliance with the list of heuristics. Every usability problem is recorded, including details about the affected heuristic, a severity rating, and optionally a solution to the problem.

These problems are then filtered, classified by heuristic, and consolidated into an evaluation report, which can be used to improve the interface design.

Groupware heuristic evaluation is likely the cheapest and fastest method for evaluating the usability of groupware user interfaces, but because it requires various experts to increase the problem detection rate, it is prone

to multiple subjective assessments, which may be hard to combine in a meaningful report. Another critique to this method is that it does little to support the analysis of problem causes, which may lead to inappropriate solution proposals (Cockton and Woolrych, 2002).

Groupware Walkthrough

This method relies on a set of expert evaluators walking through the steps that users would have to perform to carry out collaborative tasks, and answering general questions about the experience with the groupware interface (Pinelle and Gutwin, 2002). It is based upon cognitive walkthrough, a method for evaluating the usability of single-user interfaces (Polson et al., 1992), which was adapted to the multi-user case by abstracting individual tasks and focusing the evaluation on the collaborative parts of group work.

The method can be applied without a functioning system because the evaluation is grounded on textual descriptions comprising three types of information: a) the collaborative tasks, which are ultimately converted into arrangements of mechanics of collaboration; b) the recommended procedure to perform the tasks through a particular user interface; and c) the knowledge and roles expected from the users as they collaborate. This bears some resemblance with the cooperation scenarios described on page 25, but with considerable less details, to save time and reduce costs.

During the walk through sessions each evaluator can play one or more roles within the group, keeping in mind the original goal of the collaboration, the expected user population, and the recommended procedure. Any problems encountered are recorded and discussed in a meeting with all experts involved in the evaluation. Thus, as with groupware heuristic evaluation, it may be complicated to arrive at a final report based upon multiple opinions.

2.3 Discussion

Confronting the previous methods with the three levels of human activity mentioned in Figure 1.1 on page 3, it becomes apparent that groupware evaluation for tasks performed at the rational and social levels (lasting minutes, hours, days, and more) is covered by several methods, in contrast

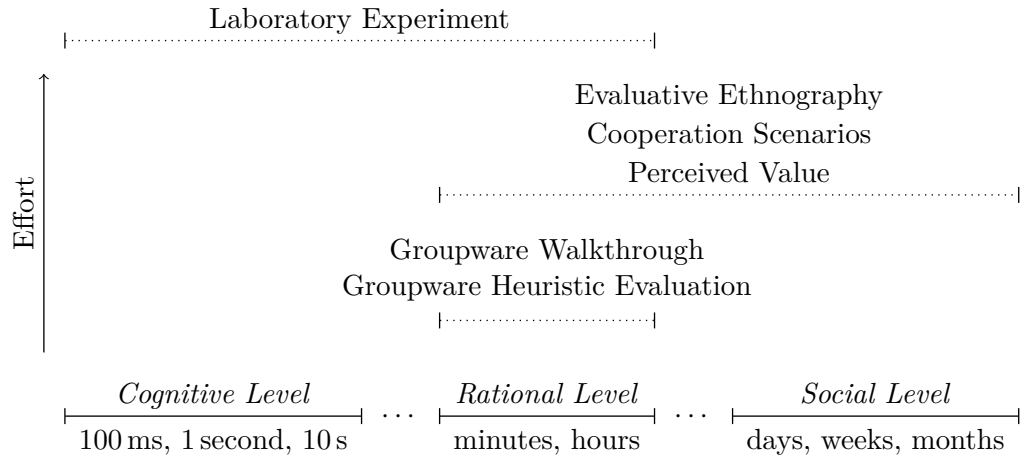


Figure 2.2: Groupware evaluation methods and levels of human action. The methods requiring the least effort to apply address computer support for tasks at the rational level, lasting minutes to hours (see lower middle region). Evaluations of more complex collaborative tasks, at the social level, consider the group and organisational contexts, and so are more expensive to conduct (upper right). The cognitive level of human action is covered by laboratory experiments, which are expensive to plan and execute (upper left region).

to the quick cognitive level tasks (from tenths of a second to seconds in duration), which are addressed in laboratory experiments (see Figure 2.2).

This means that a significant amount of effort is needed to evaluate computer support for small and yet ubiquitous collaboration tasks, because, for the reasons I mentioned earlier, experiments are expensive to plan and execute. This is illustrated, for instance, in a long series of experiments solely dedicated to minimising the disruptive effects caused by communication delays on simple tasks such as recognising gestures made with telepointers,[4] spanning a period of *six* years, in which a handful of techniques was tested (Stuckel and Gutwin, 2008; Dyck et al., 2004; Gutwin et al., 2004, 2003; Gutwin and Penner, 2002).

The other evaluation methods can be applied with varying effort but are appropriate for higher levels of human action:

- Groupware heuristic evaluations and walkthroughs are confined to computer-mediated tasks at the rational level because extending their scope to the social (with more people involved and having to cope with

4 Telepointers are replicated cursors that show the location of the mouse pointer of each user on a shared workspace, visible to all participants in a common collaborative task.

cultural influences) or cognitive levels (with finer details than those provided by the mechanics of collaboration) would go against their original purpose of reducing the evaluation complexity; and

- Perceived value, cooperation scenarios, and evaluative ethnography are suitable for assessing computer support for situated work practises, either by involving the users in the evaluation or from the immersion of ethnographers in the field. It would make little sense to divert the scope of these methods away from context-based elements and into context-free, repetitive, details. Moreover, the last two methods are expensive to apply.

From this situation, the evaluation of alternative designs for doing quick and repetitive collaborative tasks is currently being *constrained* in two ways: a) the high effort required by laboratory experiments, which hampers formative evaluations; and b) the lack of less expensive methods.

Furthermore, the impact of this constraint is growing with the increasing adoption of groupware systems, especially on the Web, where the user community is large, heterogeneous, and volatile. Thus, there is an opportunity for researching less costly groupware evaluation methods that explore the, context-independent, cognitive level of human action.

2.4 Summary

In this chapter, I presented a conceptual overview of evaluation types and how they have been applied to groupware systems, and reviewed groupware evaluation methods organised according to the context and usability trends.

I argued that existing methods assess computer support for collaborative tasks at the rational and social levels of human action, except for laboratory experiments, which also cover the cognitive level but are expensive to conduct.

Given the increasing ubiquity of groupware systems, there is an opportunity for proposing less costly methods targeting the evaluation of design alternatives for doing quick and, especially, repetitive tasks that users all around the world are executing right now. This path has long been acknowledged for the evaluation of single-user tools, which is the subject of the next chapter.

Chapter 3

The Roots of Cognitive-Level Evaluation

In this chapter, I review cognitive-level evaluation methods and show how they can be applied to predict the usability of *single-user* computer systems, which, so far, has been the prime application domain of these methods. These methods have attained considerable success but also have limitations, which I discuss, particularly regarding some approaches to apply cognitive-level evaluation in group work settings.

3.1 Background and Concepts

Cognitive-level evaluation, grounded on quick and repetitive tasks that take between tenths of a second to several seconds to execute, is well-known in the field of Human-Computer Interaction (HCI). In fact, it has matured to the point that an extensive body of psychological knowledge has been packaged into *engineering models of human performance*, that designers and evaluators can use to assess the usability of interactive computer systems (John and Kieras, 1996a; Card et al., 1983).

The original motivation for engineering models of human performance was to provide an applied information-processing psychology that designers could use to understand the factors that govern human behaviour during computer operation. These models would supply the means for doing *quick*

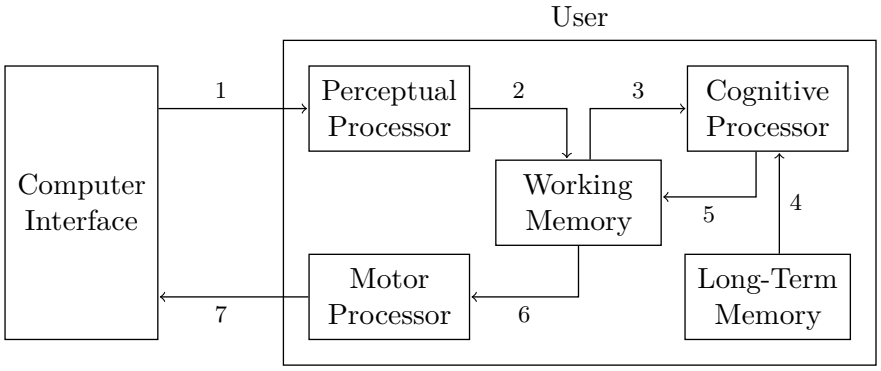


Figure 3.1: Model Human Processor, adapted from Card et al. (1983, Fig. 2.1). The user is represented as processors and memories. The numbers show a sequence of steps that reflects human information processing with a computer, from the perception of new information to the corresponding motor activity.

calculations, with good enough *approximations* of human performance, and would be based upon task analysis because much of the human behaviour is dictated by the structure of the task environment (Card et al., 1983, Ch. 1).

To this end, one of the first objectives in the development of engineering models was to simplify and combine a myriad of theories that were scattered in the psychological literature, such as Fitt's Law and the Power Law of Practise (Card et al., 1983, pp. 51–65), into what became the Model Human Processor (MHP), illustrated in Figure 3.1.

The MHP represents the user as an architecture composed of dedicated perceptual, cognitive, and motor processors, as well as working and long-term memories, each with its own functioning parameters. These components, together with general principles of operation, provide the bases for explaining human behaviour and for making predictions of human performance with a computer (Card et al., 1983, Ch. 2).

For example, the MHP predicts that the time from perception to action, useful in making micro-movement corrections that lead to our ability to, say, move the mouse pointer accurately, takes approximately 240 milliseconds. This is explained as follows (see numbered steps in Figure 3.1):

- The perceptual processor 1) converts an image from the physical world showing the current position of the mouse pointer into an internal mental representation and 2) stores it in the working memory (WM). This typically requires 100 ms;

- Then, the cognitive processor 3) recognises the new contents in the working memory, 4) chooses a corresponding action from the knowledge-base in the long-term memory (LTM), and 5) stores the selected action in the working memory. This takes about 70 ms; and

- Finally, the motor processor 6) obeys the new instructions in the working memory by 7) moving the hand that holds the mouse a little bit in the desired direction. This usually requires another 70 ms.

Although the Model Human Processor is useful in predicting the execution time for short tasks, such as the micro-adjustments in the previous example, it is difficult to use in interactive systems design because users exhibit much more complex behaviour while interacting with a computer.

This situation motivated the next developments in engineering models, which, for the sake of bridging the gap between applied psychology and design practise, concentrated on higher-level models for predicting the behaviour of *experts* executing *routine* tasks.

In these circumstances, actual performance is approximated by error-free performance, and, because no problem solving is involved, user tasks can be described via actions such as pressing keys, pointing with the mouse, locating objects on the computer screen, and others, whose durations can be derived from the MHP. These higher-level models are known as GOMS models (John and Kieras, 1996a; John, 1995; Card et al., 1983).

GOMS models represent a task through the user's knowledge of how to perform it in terms of Goals, Operators, Methods, and Selection rules.

Goals indicate what the user wants to accomplish and may contain sub-goals. For example, to edit a paragraph the user needs to locate it in the text and make the changes (a main goal with two sub-goals).

Operators are actions that the user can do to perceive or act upon the task environment and are tied to what the computer interface allows the user to do. Operators are considered indivisible, which distinguishes them from the goals, and have predetermined durations.

A *method* is a well-practised sequence of sub-goals and operators for accomplishing a goal. It reflects one way, possibly among many, in which the user stores his/her knowledge about a task. For instance, the user knows that to delete a word s/he may press the 'delete' key multiple times.

Finally, *selection rules* apply when there is more than one method for fulfilling the same goal. For instance, a rule for deleting a paragraph may be to press the 'delete' key multiple times if the paragraph is short, otherwise use the mouse to select the entire paragraph and then press the 'delete' key once. The choice of method may be a matter of personal preference or may be dictated by the user interface, but it is always expected to take place is a quick and smooth way, reflecting the essence of expert behaviour.

These GOMS concepts are still up-to-date, despite having been presented more than twenty years ago, and they still provide the foundations for new research studies. In fact, the contributions of GOMS to the body of knowledge include a family of GOMS methods for modelling tasks and predicting user performance, computer tools to facilitate the construction and validation of task models and automatically estimate user performance, and numerous success stories of GOMS applications in various contexts.

3.2 Evaluation Methods and Tools

The literature traditionally recognises four GOMS variants and respective evaluation methods (John and Kieras, 1996b): the Keystroke-Level Model (KLM) (Card et al., 1980); the GOMS method that appeared in Card et al. (1983), known as CMN-GOMS; the Natural GOMS Language (NGOMSL) (Kieras, 1988); and the Cognitive, Perceptual, Motor GOMS (CPM-GOMS) (John, 1990). These variants differ in task modelling expressivity, underlying architectural assumptions, and predictive power.

3.2.1 Keystroke-Level Model

The Keystroke-Level Model, or KLM, was first published in Card et al. (1980) and is the simplest and most practical GOMS method for evaluating user performance in human-computer interaction tasks. It is essentially GOMS without goals and selection rules (Card et al., 1983, Ch. 8).

In the KLM, a task is represented as a sequence of operators and the total *execution time* is obtained by adding together the individual operator durations. Therefore, it assumes a serial stage information processing architecture in which one operator is done at a time.

Table 3.1: KLM operators and predicted execution times, in seconds, based upon Card et al. (1983, Fig. 8.1) (left-most numeric column), Olson and Olson (1990, Fig. 7) (middle numeric column), and Kieras (2003) (right-most column).

		Time/s		
Operator	**Description**	C	O	K
`K`	Keyboard keystroke	0.28	0.23	0.28
`B`	Press or release mouse button	—	—	0.10
`H`	Home hand(s) on keyboard or other device	0.40	0.36	0.40
`P`	Point with mouse to target on a display	1.10	1.50	1.10
`M`	Mentally prepare for an action	1.35	1.20	1.20

KLM operators are identified by a letter and typically include: `K`, a keyboard or mouse keystroke; `B`, a mouse button press or release, but not both in the same movement; `H`, home the hand on the keyboard or on another device; `P`, point the cursor to a target on the display using the mouse; and `M`, mentally prepare for an action, such as searching for an object on the display. Table 3.1 shows the reference durations for these operators.

The `M` operator is different from the others because it represents non-observable user behaviour. Its presence in a task model may be hypothetical in some cases, but one important characteristic is that it is used consistently. To this end, the KLM includes heuristic rules for the placement of `M` operators, which are described in Card et al. (1983, Fig. 8.2).

To illustrate how the KLM can be applied to predict user performance, consider the following example from an object drag-and-drop task in a typical graphical desktop: the user searches for the object on the computer display, an `M`; then homes one hand on the mouse, an `H`, points the mouse cursor to the object, a `P`, and presses (but does not release) the mouse button, a `B`. The next step is to search for the place where the object is to be dropped, an `M`, followed by a `P` to move the cursor to the new location, and, finally, a `B`, to release the mouse button. The final KLM model is `MHPB MPB` and its predicted execution time is $(1.2 + 0.4 + 1.1 + 0.1) + (1.2 + 1.1 + 0.1) = 5.2$ seconds (see individual operator times in Table 3.1).

Concerning tools using the KLM, the focus has been on supporting automatic model generation based upon demonstrations with prototypes.

The CRITIQUE tool (Convenient, Rapid, Interactive Tool for Integrating Quick Usability Evaluations) is for programmers who develop graphical user

interfaces using the subArctic toolkit, and allows, via transparent modifications to the toolkit, the recording of events during prototype demonstrations, which are then used to generate the KLM model (Hudson et al., 1999).

The CogTool is for designers who create interface mock-ups with a popular Web authoring suite, which has been instrumentalised so that the Web pages, when demonstrated in a browser, automatically communicate with an external behaviour recorder that creates the corresponding KLM model (John et al., 2004).

3.2.2 Card, Moran, and Newell GOMS

The Card, Moran, and Newell GOMS, or CMN-GOMS, is the original GOMS model presented in Card et al. (1983, Ch. 5), whose building blocks are not only methods and operators, as in the KLM, but also goals, which may be decomposed into sub-goals, and selection rules, for choosing among various possible methods based upon conditional statements.

CMN-GOMS models are usually built using a top-down, breadth-first, approach, in which goals are decomposed into sub-goals, one layer at a time, until user actions can be described using operators. In the end, a goal hierarchy should be apparent, even though no explicit guidelines exist on how to represent the concepts in the task model. So, for example, operators may have different names and yet correspond to the same KLM operator, and selection rules may be described as informal text in side notes.

To predict user performance in a task, the operators' durations are added together, similarly to the KLM, thus a serial stage information processing architecture is also assumed in CMN-GOMS. However, there may be more than one way to fulfil the same task because of the selection rules. If this happens, the evaluator must traverse the goals hierarchy taking into account which methods the user would select in a particular task scenario, to then calculate the total estimated task duration.

Tool support for CMN-GOMS modelling varies in features and purpose but at least allows goal hierarchies to be created and user performance to be predicted from the operators' durations, as shown in Baumeister et al. (2000). I describe two tools from that study, namely QGOMS and CATHCI, and mention an additional tool called VISNU.

The QGOMS (Quick GOMS) tool provides a graphical tree-like visualisation of the task model and is characterised by the possibility of attaching probabilities to sub-trees to estimate how frequently they would be executed in real task executions (Beard et al., 1997).

In contrast, CATHCI (Cognitive Analysis Tool for HCI), cited in Baumeister et al. (2000), supports selection rules by querying the user about which method should be chosen for each particular task instantiation. In addition, it facilitates the building of plausible models by remembering the user to consider details such as the position of the hand prior to the placement of a typing or pointing operator.

The VISNU (Validation of Intelligent Systems aNd Usability) tool supports GOMS task modelling, as the other tools, and it also provides mechanisms for matching the models against logs of users interacting with an interface prototype (Mosqueira-Rey et al., 2004). This integrated approach is grounded on the possibility of extracting model instances from the logged data, which can then be used to confront the predicted and actual execution times and to identify the most used portions of the model.

3.2.3 Natural GOMS Language

The Natural GOMS Language, or NGOMSL, builds upon CMN-GOMS by formalising the representation of goals, operators, methods, and selection rules into a notation that resembles a computer program written in a procedural language. It also introduces new operators for accessing the long-term and working memories, which provide a more fine-grained control over internal information processing than is available through the general purpose `M` operator from the KLM (Kieras, 1988).

In NGOMSL, goals and sub-goals are either represented as methods or as selection rules. In the method form, a goal is a sequence of steps, and a step may be an operator, a sub-goal invocation, or an explicit indication that the current goal has concluded. In the selection rule form, a goal is a sequence of mutually exclusive conditions and corresponding sub-goal invocations.

Figure 3.2 shows an example of an NGOMSL model for a ‘move text’ task, which comprehends two sub-goals, ‘cut text’ and ‘paste text,’ which is also translated into a textual description in the next paragraphs.

Method for goal: **Move text**
Step 1. Accomplish goal: Cut text.
Step 2. Accomplish goal: Paste text.
Step 3. Return with goal accomplished.

Method for goal: **Cut text**
Step 1. Accomplish goal: Select text.
Step 2. *Retain* that command name is `CUT`,
and accomplish goal: Issue a command.
Step 3. Return with goal accomplished.

Method for goal: **Paste text** ...

Selection rule set for goal: **Select text**
If text is word, then accomplish goal: Select word.
If text is arbitrary, then accomplish goal: Select arbitrary text.
Return with goal accomplished.

Method for goal: **Select word**

Step 1: Determine position of middle of word.	`M`
Step 2: Move cursor to middle of word.	`P`
Step 3: Double−click mouse button.	`BBBB`
Step 4: Verify that correct word is selected.	`M`
Step 5: Return with goal accomplished.	

Method for goal: **Select arbitrary text** ...

Method for goal: **Issue a command**

Step 1. *Recall* command name, and retrieve from LTM the menu name for it, and *retain* the menu name.	`M`
Step 2. *Recall* menu name, and determine position of menu on menu bar.	`M`
Step 3. Move cursor to menu on menu bar.	`P`
Step 4. Press mouse button.	`B`
Step 5. Verify that correct menu appears.	`M`
Step 6. Move cursor to command name.	`P`
Step 7. Verify that correct command name is selected.	`M`
Step 8. Release mouse button.	`B`
Step 9. *Forget* menu name, and *forget* command name, and return with goal accomplished.	

Figure 3.2: Fragment of an NGOMSL model for the 'move text' task, adapted from John and Kieras (1996b, Fig. 4). Goals are shown in bold face, and operators for accessing the working memory in italic. KLM operators (see Table 3.1) are displayed on the right, when equivalent to the NGOMSL statements on the left.

To cut the text, the user must first select it in one of two ways, depending on the quantity of text. Assuming it is simply a word, then 'select word' is invoked, which can be be represented by a sequence of KLM operators, namely `MPBBM`, with a predicted execution time of 3.7 seconds. After completion of this goal, the 'select text' goal also concludes, and the next action from the user is step 2 in 'cut text.'

In this second step, the 'issue a command' method is called with an explicit indication that the name of the command to be issued, `CUT`, is to be stored in working memory. Next, the name of the menu that contains the `CUT` command is retrieved from long-term memory (LTM) and retained in working memory so that it can be visually searched for on the computer screen. Then, the user opens the menu and clicks on the `CUT` option. The last step in the 'issue a command' method is the removal of both the command and menu names from the working memory, meaning that these are not required anymore for the fulfilment of the main goal.

After cutting the text, the user would proceed to the 'paste text' sub-goal, which is not detailed for the sake of brevity.

The previous description shows that NGOMSL assumes a serial stage information processing architecture because task execution times were obtained from the sum of the operators' durations (for instance, in the 'select word' method), as in the KLM and CMN-GOMS.

Interestingly, NGOMSL goes beyond predicting task durations and also estimates *procedure learning time* through an approximation that takes into account the number of statements in the model and a measure of similarity between methods, which roughly translates into: more statements in a task increase the learning time, and the more consistent the user interface, the less time it takes to learn the task procedure.

The models in NGOMSL can be processed and executed by GLEAN (GOMS Language and Evaluation ANalysis), a simulation tool described in Kieras et al. (1995), for which an explanation of how to build models can be found in Kieras (2003). This tool compared favourably against two other GOMS tools because it additionally provides run-time analyses of execution time and mental workload,[1] static analyses of procedural learning times, and

1 The computation of mental workload is based upon the maximum number of items in working memory during the task simulation. See *retain* and *forget* operators in Figure 3.2.

keeps track of the user's hands to automatically insert homing operators during the execution of the model, if needed (Baumeister et al., 2000).

3.2.4 Cognitive, Perceptual, Motor GOMS

The Cognitive, Perceptual, Motor GOMS or CPM-GOMS, is the most complex GOMS method for predicting user performance in human-computer interaction tasks because it assumes a *parallel* stage information processing architecture directly grounded on the Model Human Processor (see Figure 3.1 on page 34). In other words, it dissects KLM operators in terms of cognitive, perceptual, and motor acts and accepts that these may be performed in parallel, depending on the task circumstances (John, 1990).

For example, CPM-GOMS assumes that an expert user can press a keyboard key with one hand at the same time that s/he clicks the mouse button with the other hand, which suggests that the execution time would be 0.28 seconds instead of the $0.28 + 0.20 = 0.48$ seconds that would be estimated by the KLM, CMN-GOMS, or NGOMSL, via the serial execution of the `KBB` operators (see individual operator times in Table 3.1 on page 37).

The parallelism in CPM-GOMS requires a different notation than the sequences of operators and goal hierarchies used in the other GOMS variants, for which PERT charts[2] were chosen, with the following elements:

- Perceptual, cognitive, and motor operators, such as separate left and right hand movement, eye movement, visual perception, and cognitive recognition, which may be concurrently executed if the task permits;
- Utilisation of these operators over time; and
- Dependencies between operators, to guarantee model plausibility by preventing situations in which, for instance, a motor operator would start before the required target location had been perceived by the eyes and recognised by the cognitive processor.

An example of a CPM-GOMS model for the 'move text' task depicted in Figure 3.2 can be found in John and Kieras (1996b, Fig. 7), which shows how an expert user can use both hands simultaneously to perform this task.

2 PERT (Program Evaluation and Review Technique) charts are commonly used in project management to plan and keep track of tasks and resources (Pressman, 2001, Sect. 7.7).

An important property of PERT charts is the *critical path*, which, in project management, is the chain of tasks that determines the duration of the entire project. Analogously, the task execution time in CPM-GOMS is obtained by adding the durations of the operators on the critical path.

By following this reasoning, CPM-GOMS successfully predicted that a new system would increase task duration design because the changes would introduce extra operators on the critical path, despite reducing the number of operators in other parts of the task (Gray et al., 1993).

One consequence of the fine-grained level of detail in CPM-GOMS is that the models tend to be complex and lengthy, requiring intensive labour if built by hand. This, in turn, increases the likelihood of multiple evaluators finding discrepancies in some parts of the model.

To address these issues, researchers have been documenting *templates* of micro-strategies for carrying out common tasks, such as reading text on the computer screen (John and Kieras, 1996b, Sect. 2.4) and using the mouse to point to a target on the screen and click on it (Gray and Boehm-Davis, 2000), which can be reused in many contexts.

In addition, these templates have been integrated into a computer tool called Apex, which accepts a GOMS hierarchical task description as input, converts its KLM operators into the corresponding templates, then runs instances of the task in a Model Human Processor simulator, automatically adjusting the parallelism among the operators as a function of the run-time availability of the cognitive resources, and finally produces a PERT chart as output, whose critical path determines the total predicted task duration (Vera et al., 2005; John et al., 2002).

3.3 Application Domains

There is a long string of success stories to the credit of engineering models of human performance and their use in GOMS-based evaluation methods. The usability predictions provided by these methods have been successively validated in the evaluation of real-world systems, most often in scenarios of individual work, although there have also been some applications in group work settings.

3.3.1 Individual Work

The classic application of engineering models of human performance with GOMS methods was to predict the performance of a single user doing letter and table typing, text assemblies, and similar tasks on a document that only s/he could modify (Card et al., 1983).

This early assumption that a single user is the *only* person that can manipulate objects using the options offered by the computer interface, thus reflecting individual work, became predominant in future studies.

Along these lines, John and Kieras (1996a, Sect. 3) describe GOMS-based evaluations in scenarios as diverse as computer-aided design, industrial scheduling, television control, individual training, and call centres. In this last case, the purpose of the evaluation was to decide about the adoption of a new telephone operator workstation that promised time savings. However, the usability estimates predicted a slow down of the operator performance, a situation that was actually confirmed in empirical field trials.

Additional evaluations using GOMS are outlined in John (1995, p. 85), covering the use of map digitisers, video games, airline schedules, spreadsheets, and three text editors, in line with the initial GOMS research.

Also, the GOMS tools mentioned earlier indicate more scenarios in which engineering models have been successfully applied to individual tasks, namely in conference registration (Hudson et al., 1999), radiology image processing (Beard et al., 1997), operation of automatic teller machines (Vera et al., 2005), and personal contacts management (Mosqueira-Rey et al., 2004).

Finally, recent studies have applied GOMS-based methods to evaluate the usability of mobile phones (Holleis et al., 2007) and remote robot controllers (Drury et al., 2007).

3.3.2 Group Work

Beyond the individual work domain, with a one-to-one relationship between a user and a workplace, GOMS methods have also been applied in group work settings, more precisely in scenarios where users execute a set of specified procedures using computers. To the best of my knowledge, the literature includes only two such applications, following two approaches called distributed GOMS and ‘group of models.’

Min et al. (1999) propose an application of GOMS to group work, called Distributed GOMS (DGOMS), based upon the assumption that the group task can be successively decomposed until sub-tasks capable of being performed by one person are reached. Then, the sub-tasks are independently modelled using a modified version of NGOMSL, which additionally includes a communication operator for coordinating the users' activities, whose execution time derives from informal observations.

To go back to the group level of analysis, the sub-tasks in DGOMS are arranged in a PERT chart and the dependencies between them identified. In this way, the duration of the group task can be estimated by adding together the predicted execution times of the tasks on the critical path.

Another application of GOMS to evaluate group work follows the 'group of models' approach (Kieras and Santoro, 2004). The first step is to build a separate model of user behaviour for each specialised type of interaction with the computer. In other words, each model represents a specific role that can be assigned to the users.

These models are then interconnected using a simulated coordination protocol, based upon the assumption that some external communication channel, such as a radio intercom, is used to coordinate group tasks. Finally, the models are instantiated to a specific work scenario (possibly with many users assigned to the same role), and run in parallel using the GLEAN tool, which produces a computed estimate of group performance.

3.4 Discussion

The variety of GOMS methods and the wide range of applications in various domains show that engineering models of human performance offer a sound basis for assessing the usability of computer systems.

A part of this success comes from the possibility of using GOMS at almost any stage of the software development process, thus supporting *formative* evaluation, benefiting both designers and evaluators:

- Designers can look beyond the user interface building blocks (windows, buttons, text fields, and so on) and into the cognitive factors that explain user behaviour with a computer; and

- Evaluators can predict user performance without actual users or functioning prototypes, which means that fewer laboratory experiments (expensive and time consuming) are required.

This means that more design and evaluate iterations, and more informed design comparisons, can be carried out in the same time span, which is instrumental to more usable interactive systems. Another way of putting this, is that the shorter design and evaluation cycles allow systems to be developed sooner, which is important in competitive markets.

Moreover, because no users are needed, GOMS also supports *summative* evaluation by helping deciding about the adoption of a prospective system without requiring trial tests in the field.

However, there are some fundamental *limitations* in GOMS, which have long been acknowledged by its authors (Card et al., 1980), and also analysed in Olson and Olson (1990, p. 227): the engineering models of human performance are valid for experts doing routine tasks, without making mistakes, with no interruptions, and without suffering from fatigue.

This suggests that GOMS methods may not be entirely appropriate to evaluate the usability of computer systems in some task scenarios. Of course, this situation is intrinsic to engineering models (in a broad sense), whose approximations are valid only within some well-defined boundaries.

In some cases, these boundaries can be expanded somewhat. For example, in Holleis et al. (2007) and Drury et al. (2007) the researchers opted to introduce *new cognitive-level operators* because they posited that the traditional assumptions regarding human-computer interaction (with a keyboard and mouse) were not sufficiently held with mobile phones and robot controllers. So, for instance, in the first study a need was felt for gesture and finger movement operators, otherwise the model would not be rich enough to reflect human behaviour with a mobile phone.

This path of introducing additional operators also guided the knowledge transfer between the individual and group work domains. In fact, both Kieras and Santoro (2004) and Min et al. (1999) considered an extra communication operator dedicated to group coordination. However, I argue that *existing applications of GOMS in group work settings add limited value to the design and evaluation of groupware systems*, for a number of reasons.

Firstly, the proposed communication operator abstracts the work required to coordinate the group, and yet the dynamic control of activities and data flows have long been recognised as fundamental features of groupware systems (Ellis and Wainer, 1994). So, by utilising this operator, the designer is ruling out further consideration on *how* group members can coordinate themselves using the groupware, and s/he is also missing one of the goals of groupware research which is to reduce the, sometimes high, workload imposed by complex collaborative work (Carstensen and Nielsen, 2001).

Secondly, the communication operator assumes that group coordination is carried out through explicit communication between group members, and yet users often prefer other means to align their activities with those of their colleagues without interrupting each other (Schmidt, 2002). This 'smooth' coordination can be naturally achieved by monitoring the work of others and by letting the others know what one is doing, forming what is known as group *awareness* information (Dourish and Bellotti, 1992). This critical feature of groupware systems is simply not captured by the communication operator.

Finally, the approach taken in the two applications of GOMS in group work settings assumes that each user interaction with the system should be modelled separately because all users are playing individualistic roles. However, this view of group work, in which users follow procedures, is difficult to realise and consolidate, and even harder to evaluate, in the ubiquitous collaboration scenario where users *freely* decide their next steps.

Given these circumstances, a fundamental shift is needed in the way a groupware designer or evaluator applies GOMS-based methods. It is true that individual work plays its part in group work, but to improve groupware usability it is essential to focus the design and evaluation on how users can collaborate using the functionality offered by the computer system.

3.5 Summary

In this chapter, I reviewed usability evaluation methods grounded on engineering models of human performance, known as the family of GOMS methods, which have been successfully applied since the early 1980s to predict human

performance and provide cognitive-level explanations of human behaviour with *single-user* computer systems.

I also described two applications of engineering models in group work and discussed that they offer little guidance to the design of *multi-user* interfaces because the evaluation was restricted to independent tasks executed by users playing individualistic roles, instead of being focused on the usability of collaborative tasks done through the computer interface, which, if improved, could have large net effects, as I argued in Chapter 1. This is the path that I have followed in this dissertation, beginning in the next chapter.

Chapter 4

Modelling Groupware at the Cognitive Level

Having presented the concepts and methods for evaluating the usability of computer systems at the cognitive level of human action, in this chapter I propose a model of the groupware user interface that combines human information processing and computer support for collaborative tasks.

By following this approach, I seek to expand the application of existing engineering models of human performance to groupware, where the user interacts not only with the computer, as happens in single-user systems, but also with the other users on the group *through* the computer.

4.1 Motivation

The construction of a cognitive-level model of the groupware user interface is motivated by the practical needs of designers and evaluators, who both require an understanding of the 'materials' involved in computer-mediated collaboration, namely:

- The *user*, who executes tasks by applying his/her knowledge of how to collaborate with the others; and
- The *computer*, which structures the collaboration environment by providing workspaces and the tools for interacting in them.

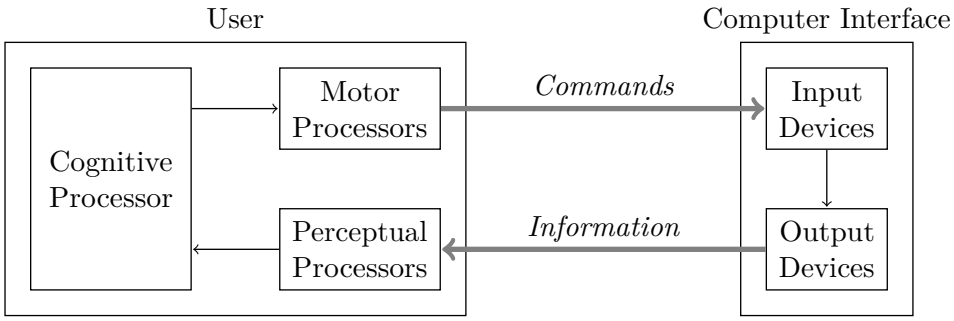


Figure 4.1: User interacting with the computer, based upon a simplified view of the Model Human Processor in Figure 3.1 and a conceptual computer interface. The motor processors command the computer via the input devices, such as the keyboard and mouse. The perceptual processors receive information generated by the output devices, such as the computer screen and speakers. More information about the input/output devices can be found in Dix et al. (2003, Ch. 2).

This user/computer duality is fundamental in the user interaction design (Dix et al., 2003, Sect. 5.2) and, particularly, is at the core of engineering models of human performance, including the Model Human Processor (MHP) and the related GOMS models (see Chapter 3).

Engineering models explain human behaviour with a computer using an architecture with perceptual, cognitive, and motor processors, which reflects user interaction with input and output devices, such as the keyboard, mouse, and computer screen. The human processors are, therefore, *linked* to the computer devices (see Figure 4.1), as in the following examples:

- The motor processors act upon the input devices, as happens when the user moves the mouse with the hand or presses keys;
- The perceptual processors capture the information conveyed by the output devices, for instance, the eyes reading text on the computer screen or the ears listening to an audio book; and
- The cognitive processor interprets the information supplied by the perceptual processors and decides which action, if any, to perform using the motor processors.

In the previous chapter I mentioned several successful applications of engineering models, but argued that existing approaches to building models of the user interacting with the computer add limited value to the design and evaluation of groupware systems, for three reasons (see also page 46):

1. Reliance on an abstract communication operator that hides the details about how users can coordinate themselves using the groupware;
2. Assumption that group coordination is confined to explicit communication, to the exclusion of other essential coordination styles, such as monitoring the work of others; and
3. Stipulation that all group members play individualistic roles by following strict procedures.

It is from this incomplete understanding of both the user and computer involvements in computer-supported collaboration that the opportunity emerges to propose a model of the groupware user interface grounded on the cognitive level of human action. To do this, I follow an approach that shifts the focus from the interactions between the user and the computer to the interactions between users, mediated by the groupware interface.

4.2 The Collaborative User

The first 'material' involved in computer-mediated collaboration is, naturally, the user, more specifically the person who is part of one or more groups or organisations. This *collaborative user* contributes to some collective effort, which means s/he requires some form of coordination with the other people on the same group, otherwise conflicts would likely occur.

Therefore, collaboration is characterised not only by the type of *group task*, such as negotiation, group decision making, brainstorming, and other types, but also by the *coordination mode* adopted by the group. For instance, a small group may rely on meetings to resolve task conflicts, whereas a large organisation may tend to establish impersonal formalised rules (Van de Ven and Delbecq, 1976).

Group tasks and coordination modes are two sides of the same coin—both are required for the group to effectively produce a combined outcome, and, more importantly, both dictate the behaviour of collaborative users. Therefore, in the following I report on reference group task and coordination mode typologies available in the literature, which will inform the proposal for the model of the groupware interface.

4.2.1 Group Tasks

There are many ways to classify group tasks, but one of the most frequently cited is the group task circumplex, described in McGrath (1984, Ch. 5).[1] This typology organises the group task space in four major quadrants, which are further subdivided into two more specific task types:

Generate These are tasks asking the group to describe how to carry out some plan of action (*planning* tasks), as well as for tasks that require the production of ideas, as in brainstorming (*creativity* tasks);

Choose These tasks require the group to find the correct answer for some problem (*intellective* tasks), or to reach consensus about the preferred decision when there is no 'right' answer (*decision-making* tasks);

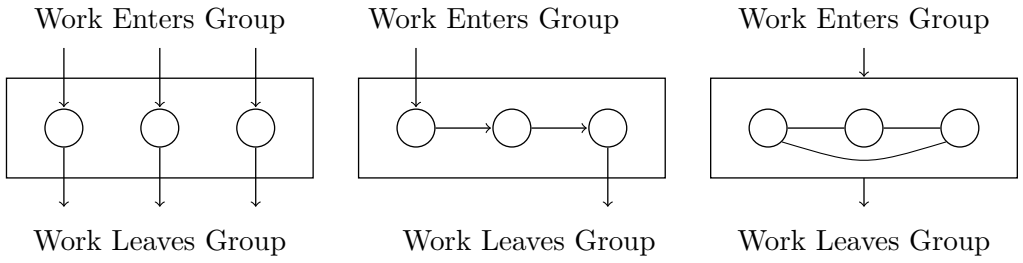
Negotiate These tasks involve making choices in conditions of intragroup conflict, caused by opposing interests, as in bargaining (*mixed-motive* tasks), or by users systematically using different preference structures or viewpoints (*cognitive conflict* tasks); and

Execute These tasks depend most on physical behaviour, such as contests or battles, in which there is a winner and a looser (*competitive* tasks), as well as performances, where the emphasis is on coordinating manual tasks, as in rowing (*psycho-motor* tasks).

Given the variety of group task types, which, at their core must have a way of dividing work among the members of the group, it is not surprising to find that multiple levels of task *interdependence* have been identified. This property represents the extent to which group members are dependent upon one another to perform their individual tasks (Van de Ven and Delbecq, 1976), comprising three increasing levels of interdependence:

1. In the *pooled* level, tasks share or produce common resources but are otherwise independent. Thus, tasks can be executed in parallel with little or no communication between the group members, which means that task interdependence is minimal;

1 Other group task classifications are also mentioned in McGrath (1984, Ch. 5), covering a period from the 1800s onwards, with emphasis on the second half of the twentieth century.



(a) Pooled. (b) Sequential. (c) Reciprocal.

Figure 4.2: Levels of group task interdependency, adapted from Van de Ven and Delbecq (1976, pp. 334–335). From left to right, the interdependencies increase: (a) each person processes a work unit with little or no contact with the other group members; (b) the work done by a person is dependent on the output of others; and (c) work is jointly carried out by the group members.

2. In the *sequential* level, tasks depend on the completion of others before beginning. In other words, tasks are done sequentially and, thus, some group members rely on others to timely finish their tasks; and

3. The *reciprocal* level has high task interdependency because group work becomes a joint effort where all the members depend mutually on each other to accomplish the task.

Figure 4.2 shows work flows within a group of three people for each level of task interdependence. Curiously, such an abstract representation makes it easier to understand that the same type of group task can be performed under more than one level. For example, creativity tasks, such as brainstorming, have been extensively studied and compared in scenarios where a pool of ideas is formed from the efforts of people working separately, corresponding to the pooled level of interdependence, versus from the concerted work of a group of people, at the reciprocal level (Dennis et al., 1994)

More interesting is the possibility of using Figure 4.2 to frame the levels of task interdependency covered by current applications of engineering models of human performance in group work settings.

In Section 3.4, I mentioned that both the DGOMS (Min et al., 1999) and the 'group of models' (Kieras and Santoro, 2004) approaches assume group members play individualistic roles, which is why they build separate models of the interaction with the computer system for each type of user. From this point on, the two approaches differ.

DGOMS uses a PERT chart to identify the tasks assigned to each group member over time, as well as the dependencies between those tasks, which are actually instances of one user passing the control to another. Thus, the DGOMS approach covers tasks with *sequential* interdependencies.

The 'group of models' approach uses a computer to generate events from a scripted scenario, which are fed to running instances of the user models, and trigger voice messages over an intercom channel. Then, the model from another user specifies how to handle a particular type of message and reacts accordingly. So, this is again a case of *sequential* interdependency.

In summary, to the best of my knowledge, engineering models of human behaviour have so far not been applied in tasks with high interdependency, and yet these tasks usually require intense, and repetitive, collaborative behaviour because multiple group members depend on one another and due to the continuous need to coordinate work.

4.2.2 Coordination Modes

The more interdependent tasks are, the more coordination they need to be effective, otherwise conflict and duplication of work may occur. In fact, *coordination* has been defined as the management of dependencies between tasks (Malone and Crowston, 1994).

Three coordination modes can be considered, with increasing information needs and more elaborate collaborative behaviour patterns. In the following I describe a typology mentioned in Malone and Crowston (1994, p. 114), with additional ideas from Van de Ven and Delbecq (1976):

1. In the *standardisation* mode, group members follow prescribed rules or plans that codify impersonal blueprints of action. This means that most work can be performed individually, as in a factory assembly line, with minimal communication with other people on the group;

2. The *direct supervision* mode corresponds to the hierarchical relationship between manager and employees, in which conflicts caused by task interdependencies are resolved by the former on a case-by-case basis. In this mode, communication is necessary and involves a small number of people, mostly in the form of one-to-one conversations; and

3. In the *mutual adjustment* mode, each group member continuously adjusts his/her behaviour to manage task interdependencies, based upon information emerging from the group, as happens in a meeting. Communication between people can be used in this mode, but because the cost of turn-taking and interruptions may become excessive, workplace monitoring is also used.

It can be seen from the above that there is minimal collaborative cost in managing dependencies with standardisation, which, of course, is the whole purpose of this mode. Thus, it is well suited for the pooled level of task interdependency, shown in Figure 4.2(a), even though rules and plans may also help, but not drive collaborative behaviour, in the other two modes.

Communication is the main vehicle for resolving task dependencies in direct supervision, which makes this mode especially adequate for sequential work, illustrated in Figure 4.2(b), where group members awaiting the output from others are informed they can proceed.

There is a collaborative cost attached to this coordination mode, which, reporting on the use of engineering models of human action in group work settings, is addressed in Min et al. (1999) by using abstract communication operators whose execution times were obtained from empirical measurements, whereas Kieras and Santoro (2004) assumes users only send or receive messages if enough cognitive resources exist.

Regarding the mutual adjustment coordination mode, its reliance on both communication and monitoring, makes it ideal for resolving task interdependencies in the reciprocal level, shown in Figure 4.2(c). In fact, the possibility of *monitoring* the work of others is one of the preferred ways to keep up and be aligned with the group, because it does not require interrupting the other group members (Schmidt, 2002).

Naturally, monitoring in the workplace is only possible because the others are publicly *displaying* some relevant aspects of their work, revealing what is known as group awareness information, necessary for the 'understanding of the activity of others, which provides a context for your own activity' (Dourish and Bellotti, 1992).

Many more definitions and variants of awareness exist, such as of the social, task, and conversational kinds (Drury and Williams, 2002, Table 1),

Table 4.1: Task interdependencies and coordination modes, adapted from Malone and Crowston (1994) and Van de Ven and Delbecq (1976), including typical collaborative behaviour that group members adopt to resolve conflicts and to keep work coordination as unobtrusive as possible.

Task Interdependency	**Coordination Mode**	**Collaborative Behaviour**
Pooled	Standardisation	Minimal communication
Sequential	Direct supervision	Communication
Reciprocal	Mutual adjustment	Communication, monitoring, displaying

but perhaps one of the most comprehensive studies about awareness is that of Gutwin and Greenberg (2002), which focuses on the *workspace* and includes:

- What information makes up workspace awareness, namely who is on the group, what is going on, where is activity taking place, and when and how did changes happen;
- How is workspace awareness information gathered, including via conversations between users and observations of body gestures, as well as how people interact with the environment and manipulate objects (see also the mechanics of collaboration in Table 2.5 on page 29); and
- How is workspace awareness useful to collaboration, such as to simplify communication, help coordinate activities, and give support for the anticipation of events.

The extensive coverage of group awareness in the literature attests the importance of this concept. In fact, as I alluded earlier, without the displaying and monitoring of awareness information groups could easily become inundated with internal communication requests just to keep everyone up-to-date. Thus, collaborative behaviour under mutual adjustment coordination is more elaborate than in the other two modes (see Table 4.1).

* * *

In summary, the understanding of the collaborative user behaviour comprehends the types of group tasks s/he performs, namely generate, choose, negotiate, and execute, *plus* the coordination modes s/he needs to engage to resolve task interdependencies, shown in Table 4.1. A successful model of the groupware interface must address these two dimensions of group work, which is the topic of the next section.

4.3 The Groupware Interface

The second 'material' in computer-mediated collaboration is the computer, more precisely the *groupware interface* that supports the work of the collaborative user.

The starting point of this exposition is Figure 4.1 on page 50, which shows a generic representation of a user interacting with a computer. From this initial setting, I describe five interaction patterns, or *information flows*, identified in the literature, and show which ones are necessary to support each of the three coordination modes in Section 4.2.2.

All information flows mediated by a computer can be manipulated using input/output devices, such as the keyboard, mouse, screen, microphone, speakers, and so on. However, to help understand the complex user behaviour inherent to the mutual adjustment mode of coordination, I propose specialised groupware *input/output devices* for the monitoring and displaying of group awareness information.

Finally, I show that the information flows and input/output devices can be combined into three popular types of *workspaces*, namely private, shared, and mixed-focus, and in doing so I synthesise my proposal to organise the design space of groupware user interfaces.

4.3.1 Information Flows

Figure 4.1 (on page 50) shows that the user and the computer can be seen as two interlinked information processors, one reacting to the other in cyclic patterns of interaction. I call these patterns *information flows* and in the following I describe five such flows that support the work of the collaborative user. As will become clear, depending on the coordination mode, different combinations of information flows will be required for effective collaboration.

Feedback

Feedback corresponds to the computer response to an action initiated by the user (see Figure 4.3). This feedback can then be used to guide future actions (Douglas and Kirkpatrick, 1999). For example, when the user clicks with the mouse on a graphical object, the computer interface alters its appearance,

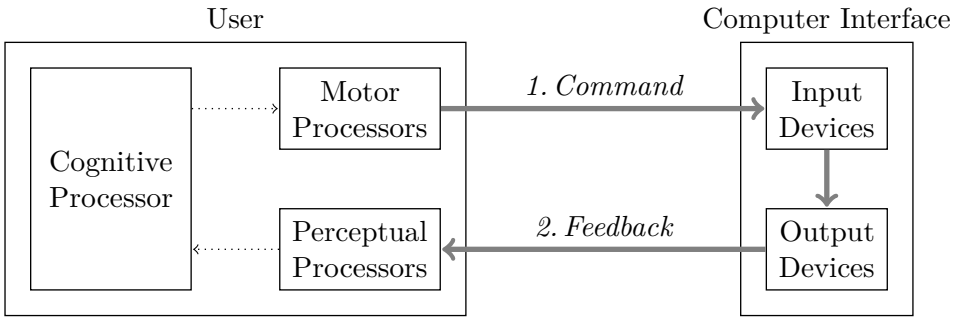


Figure 4.3: Feedback information flow: the user activates the motor processors to command the computer through the input devices (step 1) and the computer responds by sending feedback information via the output devices, which is captured by the user's perceptual processors (step 2), and may guide future actions.

usually by changing the colour of the background surrounding the object, so that it can be perceived as being selected by the user.

Feedback happens almost constantly during human-computer interaction, to the point that we find it disturbing when the computer interface does not give a reply to our commands—we expect the computer to provide feedback in confirmation to our actions.

Feedforward

The second type of flow concerns the delivery of feedforward information initiated by the computer to make the user aware of the available action possibilities (Wensveen et al., 2004). This information, in turn, may or may not trigger a response from the user (see Figure 4.4).

For example, feedforward happens when the user receives a notification stating that a large file download has completed or that new e-mail has arrived; the user may ignore this and resume what s/he was doing, or s/he may stop the current task and give priority to dealing with the recently downloaded file or to the new e-mail messages.

* * *

When feedback and feedforward are combined, they provide just sufficient computer support for the *standardisation* mode of coordination. Recalling Table 4.1, this mode requires minimal, if any, communication to resolve task interdependencies because the user follows codified plans of action carefully devised to minimise conflicts among group members.

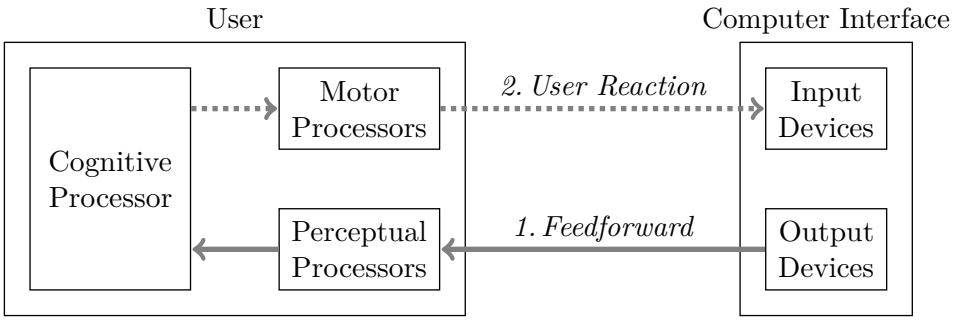


Figure 4.4: Feedforward information flow: the computer takes the initiative of generating feedforward information, which is captured by the user's perceptual processors (step 1), and this may or may not trigger a motor reaction from the user (optional step 2).

So, imagining a line worker in a virtual factory, in normal circumstances s/he is notified of the arrival of a new item via the feedforward information flow, and, as long as s/he is working (individually) on the item, only feedback information needs to be processed.

At this point, I note the feedback and feedforward are necessary but not sufficient to support the added complexity in user behaviour in the more elaborate modes of coordination. In fact, a change in perspective is needed, from the interaction between the user and the computer, to the *interaction between the users, mediated by the computer*.

This change in perspective has an immediate impact on the previous models in Figures 4.1, 4.3, and 4.4, in that the groupware interface now has multiple users connected to it, mediates all sorts of information that comes and goes between and among the users, and occupies a central position within the group, as shown in Figure 4.5.

I consider three information flows recurrently managed by groupware systems to support the higher modes of coordination, namely explicit communication, back-channel feedback, and feedthrough. These flows are outlined in Figure 4.5 and I further illustrate them in Figure 4.6.

Explicit Communication

Explicit communication addresses information produced by one user who explicitly intends it to be delivered to one or more users on the group (Pinelle

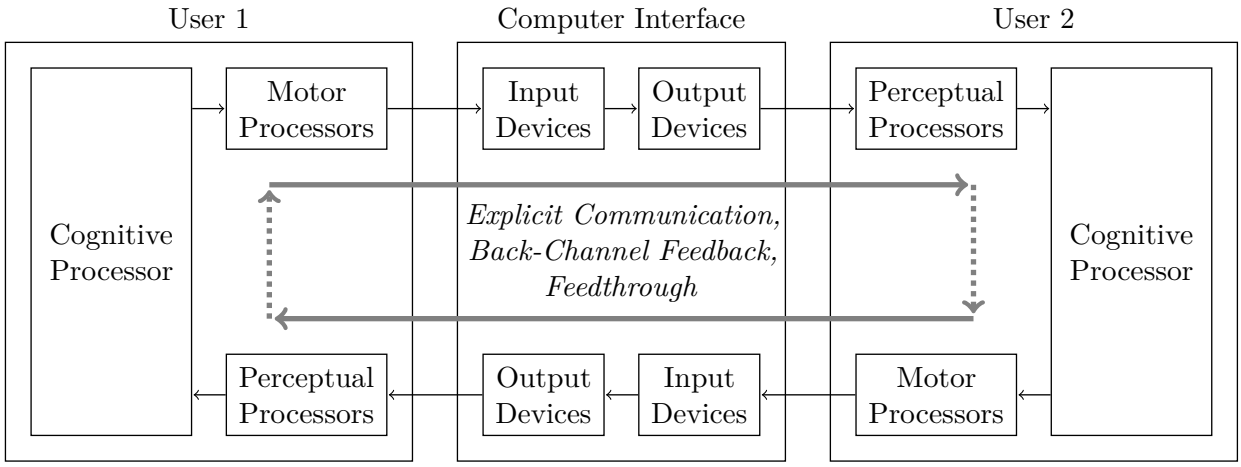
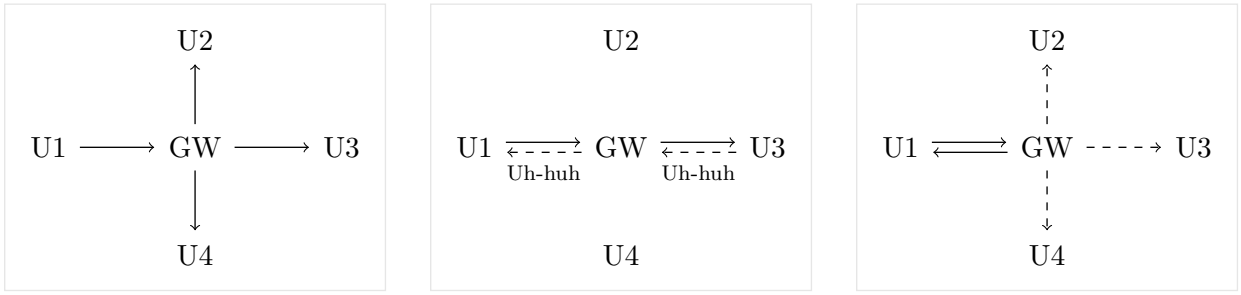


Figure 4.5: Groupware interface connecting multiple users. The groupware system is at the centre of the group and mediates various types of information, flowing between and among the users, namely via explicit communication, back-channel feedback, and feedthrough.



(a) Explicit communication. (b) Back-channel feedback. (c) Feedthrough.

Figure 4.6: Information flows between users, mediated by the computer: (a) user 1 (U1) sends a message to all other users on the group via the groupware (GW); (b) user 3 is saying utterances to indicate s/he is following the conversation with user 1; and (c) user 1 is working, and the groupware implicitly reports this fact to the other users.

et al., 2003). This happens, for instance, when a user is discussing a topic or planning some activity with other users, using speech, text, or gestures (see more examples in Table 2.5 on page 29).

This flow can be modelled by a computer interface capable of multiplexing information from input devices to one or more output devices, such as a user typing a message and the groupware sending it to the computer screens of the recipients, as shown in Figure 4.6(a).

Explicit communication, together with feedback and feedforward, are sufficient to support the *direct supervision* coordination mode (see Table 4.1),

in which a manager or supervisor resolves task interdependencies on a case-by-case basis through direct communication with small groups of people, usually in the form of one-to-one conversations.

In fact, as earlier mentioned, explicit communication can also be used in groups with coordination by continuous mutual adjustments, but because interruptions and turn-taking may become obtrusive, the groupware system has to manage additional flows, which I describe next.

Back-Channel Feedback

The back-channel feedback flow concerns unintentional information initiated by one user and directed towards another user to facilitate communication (Rajan et al., 2001). This takes place, for instance, when a listener murmurs 'uh-huh' to indicate that s/he is following the speaker, as in Figure 4.6(b), or when s/he nods the head in agreement to what the speaker is saying.

Interestingly, no conversational content is delivered through back-channel feedback, but, nonetheless, this flow is useful to regulate the exchange of ideas and avoid pauses to ask if the listener is keeping up with the speaker, that is, it serves as a coordination mechanism for negotiating actions and maintaining behavioural norms (Mark et al., 1997).

Back-channel feedback may be automatically captured and produced by the groupware based upon the users' vocal expressions and intonation variations, and also through body movements and facial expressions.

Feedthrough

Finally, the feedthrough flow concerns implicit information automatically delivered to several users reporting actions executed by one user, and is necessary to provide *group awareness* and to construct meaningful contexts for collaboration (Hill and Gutwin, 2003).

To reiterate, without feedthrough, users would have to manually notify the others, via explicit communication, on the activities being performed (Khoshafian and Buckiewicz, 1995, pp. 309–319), an overhead that would surely impair both group and individual performance.

This information flow is generated by the computer interface as a consequence of the users' inputs and is directed towards the other users on the

group. Various levels of feedthrough information may be considered, with increasing levels of abstraction:

Low-level user-interface feeds These include, for example, multiplexing keyboard and mouse movements. In the former case, a user would be able to see what others are typing, say, in a shared text document; in the latter, multiple mouse pointers would appear on the computer display, one per user (these are called telepointers);

High-level user-interface feeds For delivering information about the manipulation of user-interface elements, such as buttons, menus, input data fields, and others, which augment the information conveyed by low-level user-interface feeds. For instance, besides seeing the telepointers, users would be able to notice that a user is choosing a particular menu option; and

Application-level feeds For delivering only the events relevant to the application using filtering and aggregation mechanisms, for instance, by only delivering information that matches some user-defined criteria or by only showing complete sentences rather than individual keystrokes (as in popular instant messengers).

A simple way of supporting feedthrough is by multiplexing feedback information to several users, as shown in Figure 4.6(c). For instance, in a scenario where users use their computers to access a public virtual desktop, when a user clicks on a file icon, s/he receives feedback information indicating the file is selected, and a copy of this information would automatically be sent to the other users via feedthrough.

More interestingly, feedthrough does not have to exactly match feedback as in the previous example. It may convey different shapes and colours, for instance to reveal delays caused by communication networks (Gutwin et al., 2004), may impose changes to the delivery timings to save resources (Junuzovic and Dewan, 2009), and also the type and amount of information may be controlled by the user (Hill and Gutwin, 2004, Fig. 8).

* * *

Back-channel feedback and feedthrough complete the quintet of information flows that are necessary for effective computer support for groups using *mutual adjustment* coordination (see Table 4.2).

Table 4.2: Information flows required for each coordination mode. Mutual adjustment is the most difficult mode to support in a groupware system, because all five information flows should be in place. At the other end, standardisation only needs feedback and feedforward. Groupware for groups using direct supervision coordination must additionally support explicit communication.

	Coordination Mode		
Information Flow	Standardisation	Direct Supervision	Mutual Adjustment
Feedback	×	×	×
Feedforward	×	×	×
Explicit communication		×	×
Back-channel feedback			×
Feedthrough			×

Of course, explicit communication and back-channel feedback may be used to resolve some task interdependencies but the most important information flow in a continuous joint effort is feedthrough, because it enables group members to pick up what is going on around them and to articulate their work in conformity, without interrupting the others.

Given the significant role of feedthrough in allowing users to monitor and display group awareness information (see also Table 4.1), in the next topic I discuss input/output devices that specifically address this information flow.

4.3.2 Input/Output Devices

All information flows mediated by a computer are naturally processed by the user's perceptual, cognitive, and motor processors, and by the computer input/output devices, such as the keyboard, mouse, screen, microphone, speakers, and others (see Dix et al. (2003, Ch. 2) for more examples).

However, to help understand user behaviour in the mutual adjustment coordination mode, particularly regarding the manipulation of the feedthrough information flow, it is useful to abstract the actual physical devices, and focus on devices specialised in supporting the users' monitoring and displaying of group awareness information.

Awareness Input/Output Devices

To this end, I introduce awareness input/output devices in the model of the groupware user interface, which are specialised in *capturing* the information

users want to display to others, and in *delivering* it so that the other group members can monitor what is going on around them. Several examples of awareness devices are available in the literature, including:

Authorship lines Lines connecting objects to a participant list, to indicate authorship (cited in Gutwin and Greenberg, 2002);

Availability indicators Senses user activity to inform group members about the state of availability of each user (Begole et al., 2004);

Cursor's eye views Show the area adjacent to another user's cursor in maximum detail (Gutwin and Greenberg, 2002);

Multi-user scroll bars Multiple bars side-by-side, showing the current position of each user in the same document (Hill and Gutwin, 2004);

Over-the-shoulder views Miniaturisations of another user's main view (Gutwin and Greenberg, 1998);

Participant list Shows a list of the users belonging to the group and their presence status (Gutwin and Greenberg, 2002);

Radar views Miniaturisations of entire workspaces, showing were all users are working at the moment (Gutwin and Greenberg, 1998);

Telepointers Show the position of other users' mouse cursors, usually in distinct shapes to prevent confusion with the user's own mouse cursor (Gutwin et al., 2004);

Video images Shows moving pictures of the other users, typically focusing on the face of a single user (Dourish and Bly, 1992).

Another feature of awareness input/output devices is that they allow users to perceive the role and limitations of the computer interface as a mediator. This is especially relevant when the Internet is used to transport feedthrough information because the limitations in the available bandwidth make feedthrough delays less predictable and significantly longer than feedback delays (Stuckel and Gutwin, 2008).

Coupling Input Device

A further reason for proposing awareness input/output devices is related to a characteristic of groupware: it lets users loose the link between executed

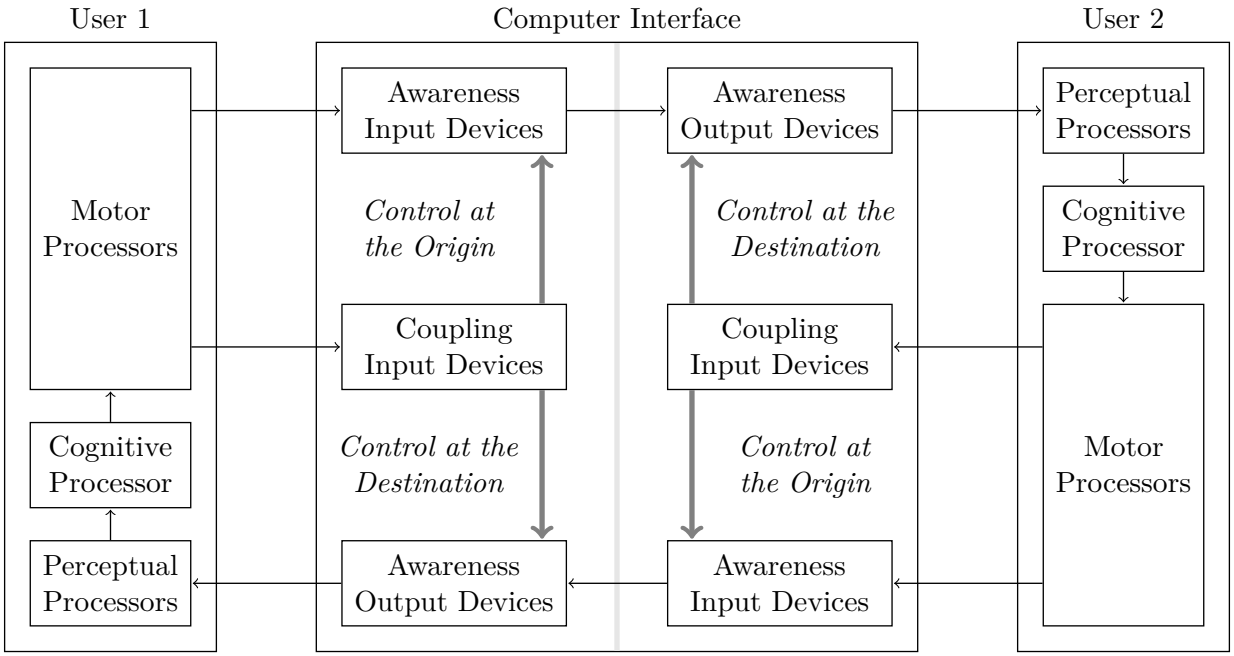


Figure 4.7: Awareness and coupling input/output devices, specialised in supporting the users' monitoring and displaying of group awareness information. The users' actions relevant to collaboration are captured by the awareness input devices. The other users can monitor that information through the awareness output devices. The coupling input devices control the type and amount of information being shared.

operations and group awareness, a situation called 'loosely coupled' (Dewan and Choudhary, 1995). Two types of coupling control may be considered:

- Users may control coupling at the origin to specify what and when information produced by or about a user should become public, and to whom. This addresses the *displaying* part of collaborative behaviour during mutual adjustment coordination (see Table 4.1).
- Coupling may be controlled at the destination to restrict the amount of awareness information that reaches the user, for example through filters on objects and types of events. So, this is useful for supporting the *monitoring* of the workplace (see Table 4.1).

In both cases the user needs cognitive-level activities to control the capture and delivery of group awareness information, and I model this with the *coupling input device*. This device and its interaction with the awareness input/output devices are shown in Figure 4.7.

The conceptual layer comprising the awareness and coupling devices, and built on top of the physical layer of conventional input/output devices, is a convenient means to discern the features specific to the various types of computer-supported workspaces, as I show next.

4.3.3 Virtual Workspaces

Groupware systems structure the collaboration environment by providing virtual workspaces and the tools for interacting in them. These are user recognisable and may be classified as follows:

- *Private* workspaces allow users to work individually with minimal or no interference from the other group members. They are suited for doing temporary work, trying out alternatives, and for working in parallel (Mark et al., 1997). In essence, this type of workspace is well suited for the standardisation coordination mode;

- *Shared* or public workspaces are similar to physical workplaces, in that users work together in a delimited 'space,' where objects and tools can be manipulated to fulfil a joint objective (Dourish and Bellotti, 1992). However, because the workspace is computer-mediated, users need not be co-located. Shared workspaces strive to support the mutual adjustment mode of coordination, as this gives group members more freedom to collaborate and resolve potential task conflicts;

- *Mixed-focus* workspaces support work scenarios characterised by users moving back and forth between doing work individually and interacting with the group, to keep up with what is going on and to share newly produced outcomes with the others (Greenberg and Roseman, 2003). This mixed-focus type of collaboration is supported not only by a shared workspace accessible to all group members, but also by private workspaces, where users can engage in individual work.

Given these descriptions of virtual workspaces, the last step in the modelling of the groupware user interface is to define them in terms of information flows and input/output devices. The result is presented in Table 4.3, which also includes the coordination modes (from which I derived the information

Table 4.3: Map of user behaviour in virtual workspaces. The design of virtual workspaces comprehends the understanding of the coordination mode adopted by the group to resolve task interdependencies, the information flows for interacting with the computer and with the other group members, and the input/output devices for controlling the information flows.

Virtual Workspace	Coordination Mode	Information Flows	Input/Output Devices
Private	Standardisation	Feedback Feedforward	Conventional
Shared or Mixed-focus	Mutual adjustment	Feedback Feedforward Explicit communication Back-channel feedback Feedthrough	Conventional Awareness Coupling

flows) presumed to be appropriate for managing task interdependencies in each type of virtual workspace.

4.4 Discussion

The model of the groupware user interface proposed in this chapter aims at representing a variety of user behaviours in collaborative environments, keeping the focus on the cognitive level of human action.

As any other model, this groupware model covers a part of reality. So, if on the one hand I mentioned a typology of group tasks, on the other hand I only derived from that the need to characterise task interdependency levels.

In turn, these levels of task interdependency lead to the identification of coordination modes that are used to solve conflicts and minimise duplication of work. These modes reflect patterns of behaviour grounded on communication between or among users, monitoring of the collaborative environment, and displaying of information relevant to the others.

In the end, I am giving less importance to group task types and highlighting the coordination modes, not only because coordination is universal, but also because a link can be established between the modes of coordination and the information flows supported by groupware systems. In this way, I could determine which flows are necessary to cover each mode of coordination.

Regarding the mutual adjustment coordination mode—the most elaborate and the preferred by users because it allows a good alignment with the actions of the other group members without interrupting them—I assumed that all groupware information flows would be necessary. However, there may be simpler alternatives that are also adequate, for instance the use of explicit communication without back-channel feedback.

In reality, the fundamental groupware information flow in the mutual adjustment coordination mode is feedthrough, which automatically generates group awareness information, without which the group could become overloaded with internal communication requests to keep everyone up-to-date.

Anyway, there may be occasions in which the quantity of group awareness information generated via feedthrough needs to be adjusted for being excessive. If this happened in a room, people would naturally regulate their monitoring and displaying behaviours. Therefore, in the computer-based case I propose that this adjustment be represented using specialised awareness and coupling devices.

Finally, the integration of the groupware information flows and input/output devices into virtual workspaces was straightforward. Again, I assumed that all information flows would be required in the more complex workspaces, but I admit some scenarios of collaboration could use less.

4.5 Summary

In this chapter, I described a model of the groupware user interface that unifies human behaviour in collaborative scenarios with groupware information flows and input/output devices.

My intention was to provide insights to groupware designers and evaluators about the design space of collaborative interactions, which can still be analysed at the cognitive level since the connection between the perceptual, cognitive, and motor processors and the computer interface was preserved.

I also posited that the model of the groupware interface can serve as a basis for expanding the applicability of existing cognitive-level usability evaluation methods, which so far have been confined to single-user interfaces. I will demonstrate how this can be done in the next two chapters.

Notes

The groupware interface model described in this chapter is an extension of work initiated by my adviser and colleagues (Antunes et al., 2005).

From that early model, I participated in the clarification of the placeholders for the awareness and coupling devices, which at the beginning appeared in both the user and the groupware interface, and which were ultimately moved to the latter. So, the idea of the user having specialised processors dedicated to group work was abandoned, in favour of the traditional Model Human Processor.

That revised model first appeared in the twelfth International Workshop on Groupware (CRIWG'06), held in Medina del Campo, Spain (Antunes et al., 2006a), and later in an article for the *Information Research* journal (Ferreira et al., 2009).

The present discussion of the model adds theoretical concepts about group task interdependencies and coordination modes, which help understand human behaviour in collaborative environments, and attest the usefulness of the proposed information flows and specialised input/output devices.

Chapter 5

Evaluating the Usability of Shared Workspaces

In this chapter, I build upon the groupware interface model introduced in the previous chapter to create a method for evaluating the usability of virtual *shared workspaces*, an essential component of groupware systems.

In line with the advantages offered by the engineering models evaluation approach, the method does not require users or functioning prototypes and is aimed at providing quantitative usability predictions, the novelty being in focusing the evaluation on tasks relevant to collaboration.

5.1 Motivation

Virtual shared workspaces (or, simply, shared workspaces) are similar to physical workplaces but they allows users to be distributed because the shared workspace and the information flows necessary for collaboration are supported by the computer infra-structure.

For example, in the physical world a chalkboard can be used collaboratively by two or more persons for drawing diagrams, annotating, making corrections, and so on, whereas in the virtual world the same persons could be in different cities and have concurrent access to a computer-supported representation of the chalkboard (see Ignat and Norrie (2006) for an overview of shared workspace groupware for collaborative drawing).

An important feature of shared workspaces is that they conceal much technical functionality from the users, namely data distribution, synchronisation, replication, security, persistence, access management, connected/disconnected modes, and other aspects.

This concealment is challenging for multiple reasons: firstly, the shared workspace must adequately bridge the gap between what the users perceive is possible to do and the underlying groupware functionality; secondly, the user interface must be at the same time compelling, innovative, easy to use, and useful; and thirdly, there is the question of how to improve the usability of shared workspaces.

The method I introduce in this chapter addresses the third challenge. Assuming that the popularity of groupware, and of shared workspaces in particular, is increasing, as I did in Chapter 1, then such usability improvements, even if small, can have *large net effects*, and may, as well, become an important factor to the success of groupware systems.

Moreover, collaboration in shared workspaces is largely defined by *quick* and *repetitive* group interaction tasks, such as giving or taking objects (see the mechanics of collaboration in Table 2.5 on page 29), which mainly depend on our perceptual, cognitive, and motor skills to be executed, in contrast with other contexts, where the emphasis may be on problem solving tasks, such as group decision making.

Such cognitive level of human action is overlooked by existing groupware usability evaluation methods (as I showed in Chapter 2) and yet the behaviour of users working on the shared workspace through the groupware interface can be approximated using engineering models.

An added benefit is that human performance at the cognitive level, such as pressing keys and pointing with the mouse, is, to a large extent, unaffected by the social, cultural, and organisational variations that exist in the community of groupware users, especially on the Web.

Finally, group work in shared workspaces strongly relies on group awareness and mutual adjustment, which are difficult to support in computer-mediated collaboration (Gutwin and Greenberg, 2002). I addressed these two features in the groupware interface model (see previous chapter) by organising the design space with information flows and specialised input/output devices, which are a fundamental part of the evaluation method.

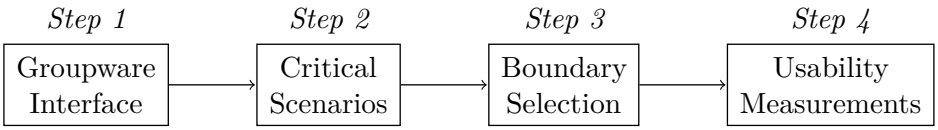


Figure 5.1: Method for evaluating the usability of shared workspaces: step 1 is to characterise the groupware interface; step 2 is for identifying critical scenarios of collaboration; step 3 is to restrict the solution space; and step 4 is for applying engineering models of human performance to predict shared workspace usability.

5.2 Method Description

The method for evaluating the usability of shared workspaces is composed of four sequential steps, illustrated in Figure 5.1.

Step 1: Groupware Interface. The method begins with a break down definition of the groupware interface in shared workspaces. This decomposition simplifies the modelling of complex groupware tools, which often organise collaborative activities in multiple intertwined spaces, usually visually recognisable, and supporting various purposes, objects, and functionality.

Using the model of the groupware interface in Figure 4.7 on page 65 as a reference, I define a shared workspace as a combination of awareness and coupling devices. I exclude any workspaces not having, at least, one awareness or coupling device, since they would not involve collaboration.

The outcome of this step is then: 1) a list of shared workspaces; 2) a description of supported explicit communication, back-channel feedback, and feedthrough information flows; and 3) a characterisation of supported awareness and coupling devices. Alternative design scenarios may also be defined in this step, considering different combinations of interface elements.

Step 2: Critical Scenarios. The second step describes the functionality associated with the shared workspaces, with a focus on *critical scenarios*, which are collaborative actions that have a potentially important effect on individual and group performance. The functionality may be decomposed into sub-actions, using a top-down strategy, but care should be taken so that the descriptions remain generic. As in the previous step, alternative design scenarios may be defined, considering several combinations of users' actions.

Step 3: Boundary Selection. This is a focusing step, in which the possibly infinite configurations of each shared workspace, including its objects and users, are reduced to particular instances according to the designer's intuition, expertise, and goals.

In this step, several characteristics of the shared workspaces may be controlled by assumptions concerning aspects such as: the position and size of graphical elements on the computer display, the awareness and coupling devices; the number of users in the group; the probabilities of user actions; the placement of objects in the workspace; and others that the designer may find relevant to workspace usability.

Step 4: Usability Measurements. The final step is dedicated to comparing the alternative design scenarios that were defined in the previous steps. These comparisons require common criteria, for which I selected the *predicted execution time* in critical scenarios of collaboration.

To this end, I utilise the Keystroke-Level Model (KLM, see Section 3.2.1 on page 36) because it is relatively easy to use and has been successfully applied in the evaluation of single-user interfaces. The application of the KLM in shared workspaces should be focused on critical scenarios having selected sequences of operators concerning quick and repetitive tasks relevant to collaboration, possibly involving more than one user at the same time.

For instance, suppose we want to evaluate the usability of several design options for managing access to objects in a shared workspace. A critical scenario occurs when a user accesses the object, immediately followed by another user trying to do the same but finding the object locked. The KLM may be used to estimate the execution times of these combined operations for each design option, thus revealing which one minimises the overall execution time. This will be discussed in one of the cases presented in the next section.

5.3 Using the Method

In this section, I apply the proposed method to evaluate the execution time in three cases of shared workspace activity: locating updated objects, reserving objects, and negotiating requirements.

5.3.1 Locating Updated Objects

The first case considers a graphical shared workspace where several objects may be updated in parallel by a group of users. An object can be a text document, a drawing, or any other type of information that is relevant to the activity of the group. In collaborative scenarios such as this it is important that users are aware of the updates that are being applied to the objects, otherwise group performance may degrade because of, for example, wrong decisions based upon outdated states of the objects, or duplicate work due to similar objects having been created elsewhere in the meanwhile.

In this case, users can play two roles: the first occurs when they update one or more objects; the second role is characterised by the need to be aware of and locate objects that have changed. I will assume that an object was recently updated, thus the evaluation will be focused on the second role. The design challenge is that there are many ways to convey object changes in a shared workspace, and some may be more usable than others.

I note that collaboration in this case is somewhat indirect, in that the focus is on information flowing from the shared workspace to individual users, although such flows are a consequence of updates made by other users. However, these information flows ease the construction of more sophisticated collaborative scenarios and, thus, their importance should be acknowledged.

Step 1: Groupware Interface. The shared workspace is capable of storing a large number of objects. However, since computer displays have limited screen resolution, access to the objects is provided by a viewport that shows only a small portion of the shared workspace. The viewport can be moved, so the whole shared workspace is effectively viewable, albeit at a cost, measured in extra execution time, that depends on the design options.

The first design uses a list of object names on the right side of the screen to provide awareness to users on objects that have recently been updated (see top left side of Figure 5.2). Because the list takes up space, the viewport is smaller than the entire display, which lowers the probability of an object being shown on the viewport at an arbitrary time. This is design scenario A.

Design scenario B features the viewport and a miniature representation of the entire shared workspace, also called a radar view (Gutwin and Greenberg,

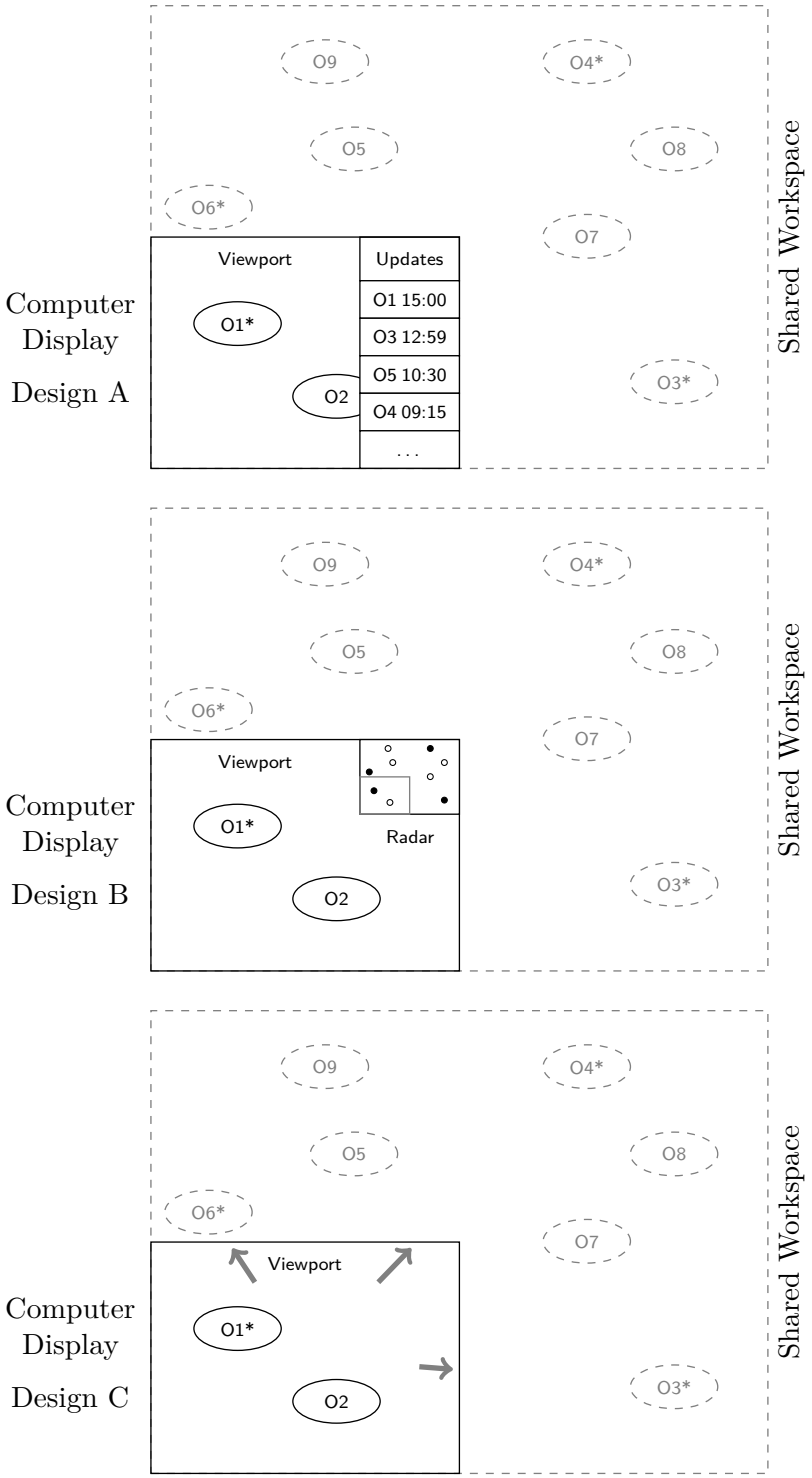


Figure 5.2: Scenarios for locating updated objects in a shared workspace. In designs A and B the computer display is partially occupied by a list of recently updated objects and a radar view, respectively, which are operated with a mouse. In design C, the entire computer display is devoted to the viewport and navigation is done with the cursor keys.

1999). Whenever an object is updated a dark-filled dot replaces the normal hollow circle on the radar, thereby making the user aware of the update (see bottom left side of Figure 5.2). As in the previous design, the radar view takes up a portion of the display space.

Finally, in design scenario C the entire computer display is devoted to the viewport. When objects are updated, and if they are not already being shown on the screen, the border of the viewport is populated with awareness indicators that look like little arrows pointing in the direction of the said objects in the shared workspace (see right side of Figure 5.2).

I assume that human input is done using a mouse with a single button in design scenarios A and B, and with keyboard cursor keys in scenario C.

At the end of this step, the groupware interface is characterised in the following terms: 1) one shared workspace stores all objects; 2) awareness is provided in the form of feedthrough information (no explicit communication or back-channel feedback is allowed); 3) awareness is supported by a viewport complemented by a list, or a radar view, or pointing arrows, depending on the design scenario; 4) there is a loose coupling between the changes that are made to the objects and the awareness that is provided to the users (an update is simply represented by an asterisk); and 5) the viewport permits coupling control, showing some objects while omitting others.

Step 2: Critical Scenarios. Regarding the critical scenario of how to locate an updated object in the shared workspace, I now describe, for each of the three design scenarios, the actions that users have to perform.

In design scenario A, the user notices that an object (which is outside of the viewport) has been updated by looking at the list of recently updated objects. To locate it in the shared workspace s/he clicks the mouse button on the corresponding item in the list, causing the viewport to be positioned in such way that the object is shown on the computer display.

With design scenario B, a dark-filled dot appears on the radar view, the user points the mouse cursor and clicks the button somewhere in the vicinity of that dot to move the viewport to that location in the shared workspace, bringing the updated object into sight.

In design scenario C, the user can navigate in the shared workspace by pressing the cursor keys. The appearance of a pointing arrow at the border

of the viewport means that an object has been updated; to know further details s/he has to follow the arrow until the object appears on the display.

Step 3: Boundary Selection. In this third step, the shared workspace is specified in practical and manageable terms, including the computer display and the viewport, upon which the performance comparison will later be based. To this end, I define the following assumptions:

1. The computer display has a typical $4:3$ aspect ratio, with a width of W units and a height of H units;
2. The size of the shared workspace is a multiple, n, of the size of the computer display;
3. The shared workspace is filled with $nW \times nH$ objects;
4. Every object has the same probability of being changed at any time.

The next two assumptions apply to design scenarios A and B, respectively:

5. The list of objects is H units high and one unit wide;
6. The radar view is approximated to a square, $n/5$ units high, rounded up to the nearest integer.

Two more assumptions apply only to design scenario C:

7. By pressing a cursor key, the viewport is moved W units (left and right keys) or H units (up and down keys) in the respective direction;
8. The opposite borders of the shared workspace are linked together, making it possible to go, for example, from the leftmost edge to the rightmost edge directly.

In these circumstances, as the size of the shared workspace increases, so does the number of objects on it and also the size of the radar view (in design scenario B), which seems reasonable. Assumption 4 is a convenient adaptation of reality as some objects, such as text documents, may be more frequently updated than others over time. Note that in design scenarios A and B the computer display is not entirely dedicated to the viewport, because the list of object names and the radar view take up some space.

Step 4: Usability Measurements. In this final step, I use KLM operators to characterise the actions that users have to execute to locate updated

objects in the shared workspace. The predicted execution time for this critical scenario will be obtained from the required sequence of operators, which depends on the design scenarios.

In all three design scenarios the estimated execution time is given by a weighted sum, $T = P_i T_i + P_o T_o$, considering two possible cases: 1) the updated object is already visible inside the viewport, with probability P_i and execution time T_i; or 2) the object is outside of the viewport, with probability P_o and time T_o.

P_i can be calculated by counting the number of objects that can be seen on the viewport and dividing this by the total number of objects in the shared workspace. P_o can be obtained by simply subtracting P_i from one.

For example, considering a computer display with $W = 4$ and $H = 3$ units, and a shared workspace with eight by six units ($n = 2$), then: in design scenario A, the list of object names takes up three units (by assumption 5), so the number of objects visible on the viewport is nine, and $P_i = 0.19$; in design scenario B, the radar view is a square one unit high (by assumption 7), giving a total of eleven objects on the viewport, so $P_i = 0.23$; and in design scenario C, the viewport uses the entire computer display, thus $P_i = 0.25$.

After obtaining P_i and P_o, I now describe the fine-grained details of how to locate an updated object using sequences of KLM operators, which will provide the T_i and T_o execution times. The sequence for the first case is only an `M` operator, from performing a visual search on the viewport and finding the object there. Since the `M` operator takes up 1.2 s (from Table 3.1 on page 37), $T_i = 1.2$ s for all three design scenarios.

The calculation of T_o, for the case in which the updated object is outside of the viewport, is done as follows: for design scenarios A and B, the user fails to find the object on the viewport, `M`; then searches the list of updated objects (or the radar view), another `M`; then points the mouse cursor to the list entry (or to the dark-filled dot on the radar), `P`; and finally clicks the mouse button, `BB` (press and release), causing the updated object to be shown on the viewport. The complete sequence of operators for design scenarios A and B is `MMPBB`, with a predicted execution time of $T_o = 3.7$ s.

Regarding design scenario C, since the updated object is initially not visible on the viewport, the user has to navigate through the shared workspace using the cursor keys, guided by the pointing arrows at the border of the

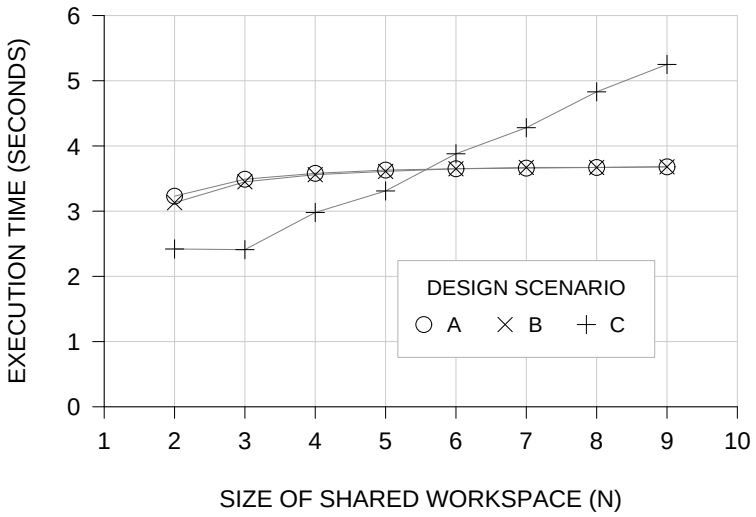


Figure 5.3: Predicted execution time for locating an updated object in a shared workspace with size n times greater than the viewport.

display (see Figure 5.2). In this case, the sequence of KLM operators depends on the size of both the shared workspace and the viewport, because the larger the portion of the shared workspace that is outside of the viewport, the more key presses are necessary to reach the object. To simplify, I assume the existence of an equation, $\overline{m}$, that calculates the average number of viewport moves to reach an updated object in the shared workspace, with unit symbol 'vm.' So, for example $\overline{m} = 1\,\text{vm}$ corresponds to one viewport move, done through a single key press (for more details on $\overline{m}$, see Appendix A).

The sequence of KLM operators for locating an updated object that is outside of the viewport, in design scenario C, can finally be expressed as an `M`, the search for the object on the viewport (and not finding it), followed by `KM`, which is a key press on a cursor key plus a visual search, repeated $\overline{m}$ times. The corresponding predicted execution time, in seconds, is given by $T_o = 1.2\,\text{s} + 1.48\,\text{s/vm} \times \overline{m}\,\text{vm}$.

At this point, the estimated execution times in the three design scenarios can be compared. The results for a display with size $W = 4$ and $H = 3$ and various sizes, n, of the shared workspace are illustrated in Figure 5.3.

In summary: 1) there is little difference in using a list of object names or a radar view to locate an updated object as the predicted execution times in design scenarios A and B are very similar; 2) the times for design scenarios

A and B rapidly converge to a maximum of 3.7 s; 3) design scenario C has a lower execution time for shared workspaces with up to five times the size of the computer display ($n \leq 5$); and 4) for larger shared workspaces, the predicted time in design scenario C increases by about 0.48 s per n.

In addition, I note that the trends displayed in Figure 5.3 are effectively independent of the size of the computer display, meaning that the graph may be seen as an easy-to-use tool, whenever the assumptions apply.

5.3.2 Reserving Objects

In this second case, I apply the method to evaluate the performance of a shared workspace that enables users to reserve selected objects. A reserved object can only be changed by the user who made the reservation; the other users have to wait for the object to be released.

In these circumstances, it is important that users be aware of which objects are currently reserved, otherwise time may be wasted in failed reservations, or work plans may be rendered inapplicable too often.

When reserving objects, users can experience one of two outcomes: a successful object reservation or a failure. The design challenge is to minimise the time wasted on failed reservations in situations where users try to simultaneously reserve the *same* object, this being the critical scenario.

Step 1: Groupware Interface. Besides the shared workspace, there are also private workspaces, allowing users to do individual work on reserved objects. However, the modelling of these private workspaces is out of scope, since the method applies to actions done on shared workspaces.

A reservation in the shared workspace is performed in the following way: first the objects are selected, and then they are dragged out of the shared workspace into the private workspace. The objects are released when they are dragged back into the shared workspace. No awareness about the state of the objects is provided to the group of users; this is design scenario A.

In design scenario B, upon a successful reservation of objects, the shared workspace displays a letter next to them, identifying the current owner. This increases group awareness and reduces inadvertent selections of already reserved objects. The letter disappears when the objects are released.

In design scenario C, while a user is selecting objects, a rectangle that comprises those objects is shown on the shared workspace. The main reason for this refinement is the production of fine-grained and up-to-date awareness information, beyond that provided by an 'after the fact' object reservation.

I assume that the user's motor activities are restricted to a mouse with a single button in all design scenarios.

In summary, the groupware interface can be defined as follows: 1) one shared workspace holds all public objects; 2) awareness information is provided via feedthrough; 3) awareness is supported by an owner letter after a reservation or by a rectangle during the selection of objects; 4) there is a loose coupling between individual work and group awareness.

Step 2: Critical Scenarios. The critical scenario occurs when users try to reserve the same object in parallel. Naturally, only one user will succeed.

In design scenario A, all objects on the shared workspace look the same, so they appear to be always available. But when users start a reservation on the same object(s) at the same time, they receive an error message, except for the one user who succeeds.

In design scenario B, users will not try to reserve objects having owner letters attached to them. However, because these letters are only shown after all steps in a reservation have been performed it is possible that two or more users try to reserve the same, apparently available, objects.

Finally, in design scenario C, besides looking at owner letters, users also see rectangles being drawn around objects on the shared workspace, meaning that other users are selecting objects, presumably to reserve them afterwards. As a consequence, users will likely choose other objects to work on.

Step 3: Boundary Selection. I make four assumptions regarding the shared workspace and the work patterns of the group of users:

1. All objects on the shared workspace are visible on the computer display;
2. Feedthrough is instantaneous (that is, there is no network delay);
3. It is unlikely that more than *two* users select or try to reserve the same object(s) at the same time;

4. The first user entering a competition for the same object(s) always succeeds in making the reservation.

Assumptions 1, 2, and 4 reduce complexity and make the analysis of the shared workspace more convenient. Assumption 3 may seem exaggerated given that all objects fit on the computer display or that the group may have many users. However, this has little importance because the reservations can, instead, be done on a large shared workspace (with the help of a viewport), without changing any of the functional details of a reservation, while making the assumption more plausible. Thus, for the usability comparison I consider two users competing for the same object.

Step 4: Usability Measurements. I now focus on the fine-grained details of how to reserve objects in a shared workspace, to the point in which this task can be described as KLM operators. It is interesting to note that the sequence of operators will be the same in all three design scenario and that the difference in execution time will be caused by the availability and timeliness of the awareness information during the critical scenario.

Regarding the KLM sequence, I assume that the user must first search for one or more objects on the shared workspace. This is converted into an `M` operator. Once the objects are located, the user moves the mouse pointer to the top-left corner of an imaginary rectangle that will encompass all objects of interest, `P`, then presses the mouse button, `B`, and moves the pointer to the opposite corner of the rectangle, `P`. The user then releases the mouse button, `B`, to complete the selection.

The last part of the reservation is done by dragging the selected objects out of the shared workspace: the user adjusts the mouse pointer so that it rests on top of one of the selected objects, `P`, presses the mouse button, `B`, drags the selected objects out of the shared workspace, `P` (no `M` operator is required because the workspaces are always in the same place), and releases the mouse button, `B`. The complete sequence of operators is `MPBPB PBPB`, which has a predicted execution time of 6.0 s.

I now compare the execution times in the critical scenario that occurs when two users try to reserve the same objects, for designs A and B.

Considering the design scenario A, the best case happens when two users start the reservation for the same object(s) at the same time. After the

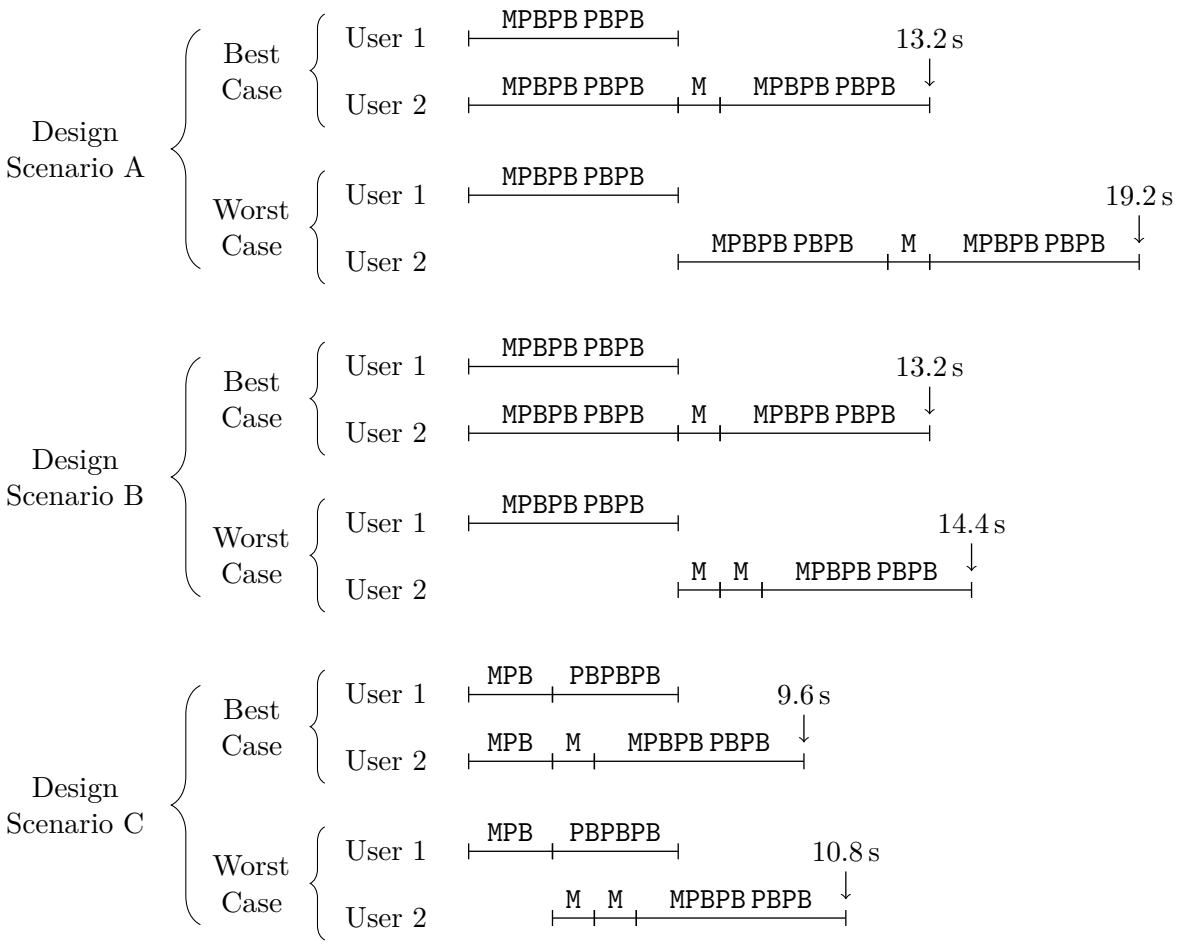


Figure 5.4: Best and worst execution times for reserving objects in a shared workspace. No group awareness is provided in design A; in design B, a ownership letter is attached to reserved objects; in design C, awareness is produced as the objects are being selected, even before the reservation is complete. The more awareness is provided, the more usable the shared workspace becomes.

6.0 s needed for a complete reservation, the second user (see assumption 4) sees an error message (an `M` operator) and starts again with another object, which takes another 6.0 s. The best execution time is then 13.2 s. The worst case happens when the second user begins just after the first finishes a reservation. Since no awareness information is provided, the total execution time increases to 19.2 s (see design scenario A in Figure 5.4.

Before progressing to the next design scenario, I highlight the following point: I assume that the time to notice and interpret an error message is equal to the duration of the `M` operator. It may be argued that it is necessary to test this assumption using laboratory experiments, as others have done

for specific tasks (Olson and Olson, 1990). However, my decision is based upon the 'simplifying logic in the KLM' and, indeed, 1.2 s has been assumed before as the duration of generic visual perception operators (Kieras, 2003).

For design scenario B, the best case is identical to that of scenario A. However, the execution time for the worst case is significantly reduced because the second user can interrupt an ongoing reservation as soon as the owner letter is displayed on the shared workspace. This situation can be modelled with two `M` operators: the first is the initial `M` of any reservation and the second `M` is for interpreting the critical situation. The total execution time for the worst case is now 14.4 s (see design scenario B in Figure 5.4).

In design scenario C, awareness information is provided as soon as an object selection starts, that is, after a sequence of `MPB`, instead of the `MPBPB PBPB` needed for a reservation. In these circumstances both the best and worst cases benefit from reduced execution times (see design scenario C in Figure 5.4). If the two users start the reservation at the same time, then at about 2.4 s they both see their simultaneous selections on the shared workspace. Then, the second user (by assumption 4) stops the current selection and starts another one, an `M` followed by a new reservation, taking a total of 9.6 s. The worst case takes 10.8 s, analogously to the worst case for scenario B, except that awareness supplied by the owner letter upon a full reservation is now awareness provided by the selection of the first object.

In summary, the method brought quantitative insights about the role of awareness information conveyed via feedthrough flows in group work support, predicting that design scenario C is faster than B by 3.6 s, and that B is faster than A by about 4.8 s, but only in the worst case scenario.

5.3.3 Negotiating Requirements

In this third case, I demonstrate the application of the method to an existing groupware tool that supports collaborative software quality assessment, using the Software Quality Function Deployment (SQFD) methodology (Haag et al., 1996). The objective of this tool is to facilitate the SQFD negotiation process by providing mechanisms in a same-time, different-place mode.

The starting point here is a previous experiment with the tool that gathered data in questionnaires and reported some usability problems, namely

that it was considered difficult to use. Further details about this tool and about the previous evaluation can be found in Antunes et al. (2006b).

Step 1: Groupware Interface. The tool has two shared workspaces: SQFD matrix and 'Current Situation.' The SQFD matrix allows users to look over a matrix of correlations between product specifications and customer requirements, as well as to observe which correlations are under negotiation. Limited awareness information is provided by the matrix, but there is a coupling mechanism that allows users to look into and modify a cell.

This coupling mechanism leads users to the 'Current Situation,' where they can observe the negotiation state in detail, including the proposed correlation, positions in favour or against, and supporting arguments, and, ultimately, express or update his or her arguments and positions. I briefly characterise the two shared workspaces in terms of awareness input, output, and coupling input devices in Figure 5.5 and Figure 5.6.

The digits inside cells in Figure 5.5 represent correlations between customer requirements (listed on the left) and product specifications (top of the matrix), going from 'weak' (`1`) up to 'strong' (`9`). When consensus is not verified for a particular cell, the digit is replaced by the symbol `?`, and, in more extreme cases, by an `F` or an `L`, meaning that a user has issued a firm position or locked the cell, respectively.

For the purpose of this case, I focus on the situation in which a negotiation is in progress. The design challenge is to minimise the time needed for a user to express or update his or her position to help the group reach a faster consensus about a particular cell.

Step 2: Critical Scenarios. In this analysis, I assume the user has arrived at the 'Current Situation' shared workspace with the purpose of examining the negotiation state in detail. As currently implemented by the tool, this information is hierarchically organised, showing: a) the product specifications and customer requirements under negotiation; b) the currently proposed correlation; c) positions in favour and against the currently proposed correlation; and d) arguments supporting positions in favour or against. This is design scenario A.

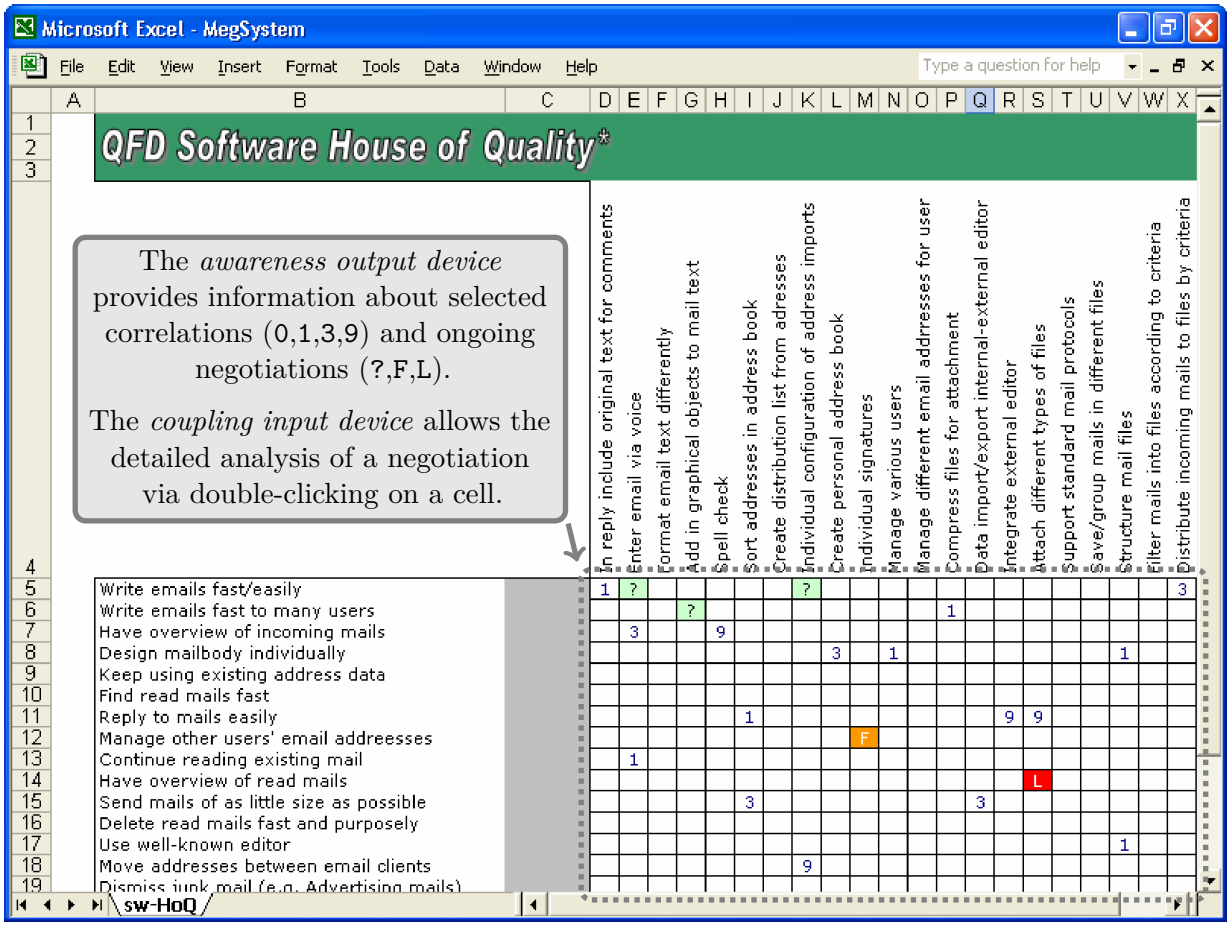


Figure 5.5: The SQFD shared workspace.

An alternative design scenario B considers a variation in the way status information is shown to the user. I assume that users assign importance to aggregate information about the number of positions against or in favour, neglecting positions where there is a clear push towards one side or the other, and analysing arguments in detail only when positions are balanced.

The selected critical scenario considers the proposal, by a user, of an alternative correlation value in 'Current Situation,' after having examined the negotiation state. This is a critical scenario because it reflects a core and repetitive activity during the negotiation process, therefore it influences individual and group performance.

I also consider a variation in the number of users involved in the negotiation process. The 'Current Situation' displays the positions and arguments

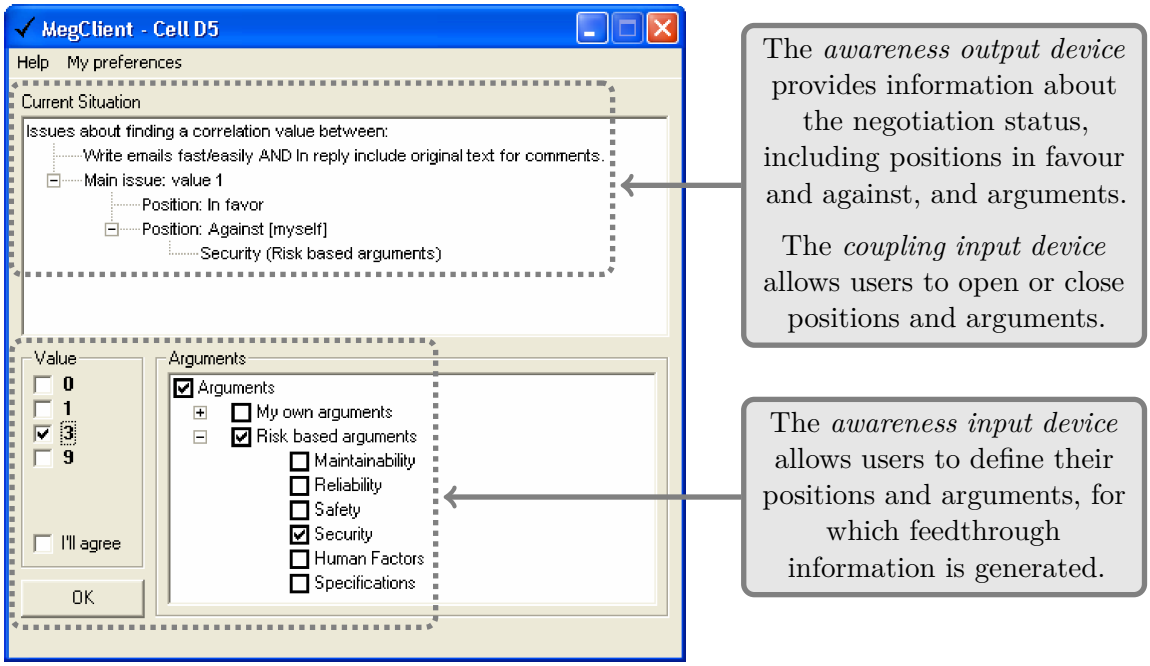


Figure 5.6: The 'Current Situation' shared workspace.

for up to three users (see Figure 5.6). Beyond this number, the user has to scroll down the window to completely analyse the situation.

Step 3: Boundary Selection. Since this case is based upon the improvement of an existing tool, the design space for the shared workspace and its operation by the users has already a practical and manageable dimension. However, given the nature of design scenario B, I consider the following three additional assumptions:

1. Users assign importance to aggregate information (see description in the previous step);
2. Two conditions, with three and six users, form the critical scenario;
3. The probability of having unbalanced positions is 25 %.[1]

Regarding assumption 2, I assume that having more than six users negotiating the same cell is a rare event, which merits no further analysis.

1 This is an approximation based upon the probability of an absolute majority with three voters, that is, having all in favour or against, two out of eight combinations, or 25 %.

Step 4: Usability Measurements. In design scenario A, with three users, the sequence of KLM operators is given by: the interpretation of the negotiation status, M, followed by a decision, M, which is expressed by the selection of a check box using the mouse, a PBB, and pressing the 'ok' button, PBB (no M operator is required to locate the 'ok' button because it is always in the same position within the 'Current Situation' window). This gives M MPBB PBB, which has a total execution time of 5.0 s. With six users, the execution time increases to 8.6 s, corresponding to MPBPB M MPBB PBB, in which the MPBPB operators are related to scrolling.

I note that some M operators considered in the previous paragraph may extend beyond the routine tasks typically modelled by the KLM. For instance, the first M in the M MPBB PBB sequence is associated with the interpretation of the negotiation status, which is significantly more complex that the selection of a check box modelled by the second M.

However, the intention in this case is not to obtain precise time values but to compare various sequences of operators in a *consistent* way across several alternative designs. Of course, there is a risk associated with modelling more complex cognitive tasks with a single M, which has to be understood and assumed by the designer, but the assumption is that this risk is equally distributed among the alternative designs, so they can still be compared.

Considering design scenario B, two situations may occur: either the positions are balanced, due to a tie or a simple majority, or they are unbalanced because of an absolute majority. In the unbalanced case, I assume the user will neglect arguments and, thus, the sequence of operators is M MPBB PBB (5.0 s to execute), similar to the previous scenario with three users.

In the balanced case, the user will analyse the positions in detail by first interpreting the negotiation status, M, followed by the opening of the list of favourable arguments, PBB (no M operator is needed to locate the list of favourable/against arguments because these are always in the same position), and corresponding analysis, M, upon which the list is closed, PBB, to give room for the opening and interpreting of the against arguments, PBB M, so that, finally, the decision is made, M, a check box is selected (with the user's decision), PBB, and the 'ok' button is pressed PBB. The total execution time for the balanced case, M PBB M PBB PBB M MPBB PBB, is, then, 11.3 s. I note that these measures apply to the scenarios with three and six users.

I also assume that the probability of having unbalanced positions is 25 % (see assumption 3). Hence, in these circumstances, the average execution time for scenario B is about $0.75 \times 11.3\,\mathrm{s} + 0.25 \times 5.0\,\mathrm{s} \approx 9.8\,\mathrm{s}$, which is higher than scenario A for both three and six users.

In summary, design scenario B may be better than or equal to scenario A, but there is a 75 % probability that it is worse than scenario A, which severely penalises the overall appreciation of design scenario B.

5.4 Discussion

The three cases, 'locating updated objects,' 'reserving objects,' and 'negotiating requirements' heavily depend on shared workspaces to orchestrate multiple users accomplishing collaborative tasks. The design of these workspaces is, thus, critical to the overall group performance. The method described in this chapter provides a common criterion to evaluate shared workspace usability, namely, execution time in critical scenarios, and allows designs to be compared to predict which features offer the best performance.

The three shared workspaces evaluated in this chapter are quite distinct. In the 'locating updated objects' case, the differences in execution times were due to the alternative ways of manipulating a coupling mechanism, a viewport, to navigate through shared workspaces with varying sizes. In the 'reserving objects' case, the focus was on the availability and timeliness of awareness information, to evaluate the best and worst execution times in environments where users act opportunistically. Finally, in the 'negotiating requirements' case, the evaluation targeted how a coupling mechanism could be designed to conserve individual cognitive effort. Taken as a whole, the method contributed to *formative* evaluation, and offered indications about the potential performance of users working with shared workspaces.

The method has three important limitations that have to be considered. Firstly, it inherits the limitations of engineering models of human performance (see Section 3.3 on page 43): it is valid for expert behaviour in routine tasks, assumes that no errors are made, that all users are entirely committed and willing to collaborate, and that users do not get tired. This may not happen in practise, but, on the other hand, the success demonstrated in

many applications of engineering models in single-user interfaces suggests that these human characteristics are evenly distributed among several design alternatives, so the comparisons still make sense.

Secondly, it assumes a narrow-band view about collaboration, restricted to shared workspaces and their mediation roles. This contrasts with other groupware evaluation methods (see Chapter 2), which offer a wide-band view about collaboration, encompassing, for example, coordination policies, influences from contextual factors, and learning, as well as more complex types of collaboration, such as group decision making. However, the trade-off to ponder is that the method restricts the view to increase the detail about the mediating role of shared workspaces and to allow quantitative comparisons of design alternatives. This restricted view has ample justification in contexts in which shared workspaces are heavily used, even when users perform intellective tasks, such as in the 'negotiating requirements' case, where users apply their expertise to evaluate software quality but are still required to repetitively operate the tool.

Thirdly, the method is somewhat limited by the selection of the critical scenarios, as they should have sufficient impact on the overall collaborative task to deserve a detailed evaluation. I posit that the bias that may exist in preliminary phases of the design process, induced, for instance, by new technologies or lack of knowledge about the collaborative context, may be reduced by applying the method at a later time. This happened in the 'negotiating requirements' case, in which usability problems identified by the users of an existing tool guided the selection of the shared workspaces and the critical scenarios. This possibility of using the proposed method in tandem with other evaluation methods, such as field studies, laboratory experiments, or heuristic-based methods (see Chapter 2), gives designers and evaluators an additional instrument of action to rely on.

5.5 Summary

In this chapter, I described a method for predicting task execution time in critical scenarios of collaboration done via shared workspaces, and showed how it can be applied to compare alternative designs in three distinct cases.

The method is grounded on engineering models of human performance, so it does not require users or functional prototypes, and is also based upon the model of the groupware interface, to focus the evaluation on the specific parts of the interface that involve collaboration.

My aim with this method is that it becomes a practical instrument for doing groupware evaluations, affording quick measurements and calculations, which drive usability optimisations, and complementing the perspectives and outcomes provided by other evaluation methods.

Notes

An earlier version of this method, with only three steps, was presented at the twelfth International Workshop on Groupware (CRIWG'06), held in Medina del Campo, Spain (Antunes et al., 2006a). At that time, the method was applied to evaluate the usability of two of the three shared workspaces described in this chapter.

Later, I added a fourth step to the method by making the boundary selection explicit, updated the two previous evaluations in conformity, and used the method to predict the usability of a third shared workspace. The results of this research were published in the *Information Research* journal (Ferreira et al., 2009).

Chapter 6

Evaluating the Usability of Mixed-Focus Workspaces

In this chapter, I present a method for evaluating the usability of groupware systems that specifically addresses computer support for *mixed-focus* collaboration, in which users alternate between doing individual work in private workspaces and interacting with the group in shared workspaces (see also Section 4.3.3 on page 66). This usually originates conflicts between the goals of the users and the goals of the group, which have to be pondered in the design of the groupware interface.

In contrast with the previous method (see Chapter 5), the evaluation developed here extends to tasks done in private workspaces because the intertwinement between individual and collaborative tasks that characterises mixed-focus collaboration influences the usability of the groupware system.

6.1 Motivation

Group work is often characterised by users moving back and forth between doing individual work and interacting with the group (Greenberg and Roseman, 2003). To address this mixed-focus type of collaboration, several groupware systems provide support not only for a shared workspace accessible to all users on the group, but also for private spaces, where users can engage in individual work.

For example, in the ShrEdit text editor users may write directly in the joint document as long as not in the same region where another user is already working on, and there is also the option of writing text in a private workspace and then moving it to the joint document (Dourish and Bellotti, 1992). Mixed-focus workspaces are also popular in Single Display Groupware (SDG) systems, in which users gathered in the same room interact with a large public display and also have access to private workspaces (Sugimoto et al., 2004; Scott et al., 2003, pp. 165–166).

An important challenge in the design of mixed-focus workspaces is that the shared and private workspaces are driven by different requirements, often reflecting conflicting goals for users as *individuals* versus users as a *group.* For instance, a distributed group benefits if all users see the same part of the shared workspace since this reduces the time spent with coordination tasks, but individual users gain from being able to navigate freely in the shared workspace (Gutwin and Greenberg, 1998). This means that improvements in the usability of mixed-focus workspaces often favour one side at the expense of the other, making it difficult to evaluate the overall result.

Another difficulty in evaluating the usability of mixed-focus workspaces is that tasks performed individually in private workspaces are usually not self-contained, that is, they often depend on the completion of tasks executed by other users and may also restrict the work of others. In the same way, the outcomes produced by a user will allow other users to advance. This task intertwinement is characteristic of 'concerted work,' in which a continuous combined effort is required from all users on the group so that a common goal is reached (Nunamaker et al., 1996). In these circumstances, the usability of the mixed-focus workspace is variable because some interaction tasks may not be possible as long as other tasks are being fulfilled.

The two previous issues, namely the trade-offs between user and group goals and the difficulties caused by task intertwinement, are addressed by the method that I describe in this chapter. As with the method in Chapter 5, the evaluation is grounded on critical scenarios of collaboration and execution times are predicted by engineering models of human performance. The main difference is that, in this case, the scenarios are evaluated not only for how much time they take but also for their contributions to group progression towards the common goal.

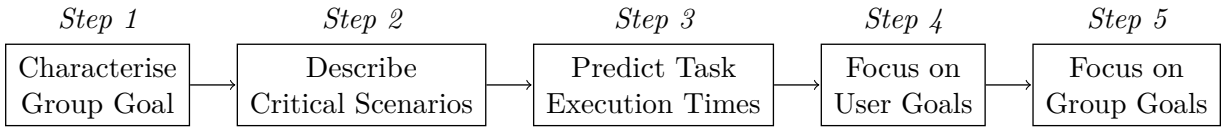


Figure 6.1: Method for evaluating the usability of mixed-focus workspaces: step 1 is for characterising the group goal; step 2 is to describe critical scenarios comprising tasks for doing individual work and for interacting with the group; step 3 is for predicting task execution times using engineering models of human performance; step 4 is to focus the evaluation on the user goals; and step 5 is for evaluating how much tasks contribute to the fulfilment of the group goal.

6.2 Method Description

The method for evaluating the usability of mixed-focus workspaces is composed of five sequential steps, illustrated in Figure 6.1.

Step 1: Characterise the Group Goal. The method begins with a characterisation of the common goal that the group needs to accomplish. This is done in terms of a conceptual metric called *goal unit,* or gu. The meaning of a goal unit depends on the context but it should be based upon an analysis of atomic contributions that users as individuals can give to the group. Thus, the group goal is defined by a quantity of goal units.

Step 2: Describe Critical Scenarios. In this step, critical scenarios of collaboration are described using text. As in the previous evaluation method (see Section 5.2 on page 73), the choice of critical scenarios is based upon the potential effects on individual and group performance. In addition, each scenario must reflect an effective contribution to the group goal, which means that it comprises tasks done in the private as well as in the public workspaces. Furthermore, tasks that reflect individual work should be distinguished from tasks involving collaboration, usually related to group coordination.

Step 3: Predict Task Execution Times. This step is for estimating execution times for each individual and collaborative task that comprises the critical scenarios. This can be done with the Keystroke-Level Model (KLM) (see Section 3.2.1 on page 36) by converting the textual descriptions into sequences of perceptual, cognitive, and motor operators, which specify the fine-grained details of user behaviour with the groupware interface.

Step 4: Focus on User Goals. In this step, the evaluation proceeds with an analysis of critical scenarios that contrasts the proportion of time for doing individual work in the private workspace with the proportion of time for interacting with the group. Alternative scenarios may be considered and it is likely that, from the user point of view, the preference goes to scenarios with lower collaboration overhead, as this increases individual productivity. However, because tasks are intertwined in a mixed-focus workspace, usability is not only in the eye of the user. This is explored in the next step.

Step 5: Focusing on Group Goals. In this final and most complex step, the critical scenarios of collaboration are evaluated against three orthogonal usability dimensions: 1) production of goal units; 2) creation of new goal opportunities for the other users; and 3) restrictions to the work of the other users while the current scenarios is being performed.

The *productivity* dimension measures the number of goal units produced per time unit, or gu/s or gu/min. The greater the value, the faster the group *may* progress towards the common goal. In individual work this dimension is enough to assess how fast a complex task can be fulfilled. However, group productivity in a mixed-focus workspace is not simply a combination of individual productivities and this is captured in the next two dimensions.

The *opportunities* dimension is related to the intertwined nature of mixed-focus collaboration: if a user stops, then the group will likely slow down, and eventually stop without reaching the common goal. This suggests that collaboration is bound by opportunity dependencies created by the achievement of user goals. The measurement unit for this dimension is new goal unit opportunities potentially created per time unit, or gu/s or gu/min. The greater the opportunities, the faster the group may progress.

The *restrictions* dimension reflects a possible negative consequence of coordination in mixed-focus workspaces: the prevention of conflicts and duplicate efforts (positive outcomes) may slow down or even impede the work of other users. Restrictions are measured in inaccessible goal units *times* the duration of the critical scenario, or gu s or gu min. This unit of measurement emphasises fast and unobtrusive execution of individual tasks: the greater the restrictions value, the slower the group may progress, because users will probably spend more time waiting to proceed.

These three dimensions provide a more comprehensive view of the usability of a mixed-focus workspace as they go beyond task execution time and link it to group progression towards the common goal. For example, a critical scenario may take the same time to fulfil in two design alternatives, but if there are differences in the opportunities and restrictions, these will determine which alternative is more usable.

6.3 Application in a Collaborative Game

I now show how the method can be applied to evaluate the usability of a collaborative game, where multiple players draw either vertical or horizontal connections between adjacent pairs of points in a matrix-like board.

The game is over when the board is filled with connections, but players must observe this rule: to be able to draw a connection between two adjacent points there must already exist a perpendicular connection between one of those points and a third point. For instance, if player Sophie is an expert in drawing vertical connections then she must consider adjacent pairs of points that contain, at least, one horizontal connection to a third point. The behaviour of an expert in horizontal connections, say, Charles, is analogous.

For illustration purposes, the board is characterised by a matrix-like arrangement of contiguous cells, numbered 1 to 9, and by an initial state that contains at least one horizontal and one vertical connection lines (see Figure 6.2). The game features a shared workspace for displaying a public view of the board and private workspaces where players can connect cell points. All user input is done using a mouse with a single button.

In order to connect points, players must first *reserve* them by selecting and dragging the corresponding cell into the private workspace. Later, the modifications on the cell will be made public when the cell is dragged back to the shared workspace, which *releases* the cell.

To minimise inadvertent selections of reserved cells, the shared workspace provides awareness information by showing a letter next to the cell number, that identifies the current owner (see letters ‘C’ and ‘S’ next to cells 2 and 4 in Figure 6.2). Additionally, the game impedes concurrent reservations of the same pairs of adjacent points. For example, if two players select the

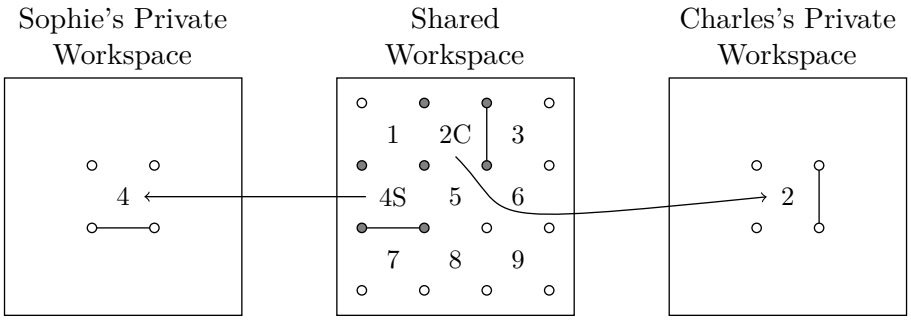


Figure 6.2: Mixed-focus collaboration game. The game board is shown in a shared workspace and players drag parts of the board (called cells) to private workspaces, where they can draw connections between points. While this happens, cells in the shared workspace are annotated with an indication of the current owner and cannot be dragged to another private workspace. After a cell has been worked it is dragged back to the shared workspace to make the updates visible to the group.

same cell or vertically or horizontally neighbour cells, and simultaneously try to reserve them, then only one player will accomplish the cell reservation, while the other is notified that the cell cannot be reserved.

It is expected that the cells remain reserved for a small amount of time due to the players' expertise and their eagerness to fulfil the common goal.

In this collaborative game all players must perform in harmony to quickly connect all pairs of adjacent points: the more horizontal connections exist, the more vertical connections can be drawn, and vice versa. Conversely, if one player stops drawing connections, the other players will have to proceed slower and may even have to stop too. In other words, the tasks executed by the players tasks are intertwined.

6.3.1 Evaluating the Initial Design

The previous paragraphs describe the initial design of the collaborative game and contain enough detail to apply the usability evaluation method.

Step 1: Characterise the Group Goal. The game ends when all points are connected with vertical and horizontal lines. For this to happen, players have to gradually draw connection lines, so a goal unit in this case corresponds to one connection line. Considering the board shown in Figure 6.2, the group goal is that all 24 connections are drawn, which gives a total of 24 gu.

Table 6.1: Textual description of tasks in the critical scenario. The RESERVE and RELEASE tasks require interacting with the shared workspace. DRAW applies to both horizontal and vertical connection lines, and is entirely done in private workspaces.

Task	Type	Description
RESERVE	Collaborative	1) Locate cell in shared workspace; 2) point mouse cursor to cell; 3) press mouse button; 4) point mouse cursor to private workspace; and 5) release mouse button
DRAW	Individual	1) Locate cell point in private workspace; 2) point mouse cursor to cell point; 3) press mouse button; 4) point mouse cursor to adjacent point in cell; and 5) release mouse button
RELEASE	Collaborative	1) Locate cell in private workspace; 2) point mouse cursor to cell; 3) press mouse button; 4) point mouse cursor to shared workspace; and 5) release mouse button

Step 2: Describe Critical Scenarios. There is one critical scenario in the collaborative game, which comprises three separate tasks: 1) dragging a cell from the shared to the private workspace; 2) drawing connection lines between points in the cell; and 3) dragging the cell back to the shared workspace. The tasks are identified by the names RESERVE, DRAW, and RELEASE, and are described in detail in Table 6.1, which also distinguishes between the individual or collaborative nature of each task.

Step 3: Predict Task Execution Times. To estimate the time needed to execute each of the tasks in Table 6.1, the textual descriptions have to be converted into sequences of KLM operators (see list of operators and corresponding execution times in Table 3.1 on page 37). For instance, the sequence for the RESERVE task is obtained as follows: in step 1 the player locates a cell in the shared workspace, an M, then points the mouse cursor to the cell in step 2, a P, and presses the mouse button (step 3), a B; in step 4, the player moves the mouse pointer to the private workspace, a P (no M operator is needed because the private workspace is always in the same place), and finally the mouse button is released in step 5, a B. The total predicted execution time is calculated by adding together the individual times, which for MPBPB gives $1.2 + 1.1 + 0.1 + 1.1 + 0.1 = 3.6$ seconds. The KLM representation for the DRAW and RELEASE tasks is also a MPBPB, hence the execution time is the same.

Table 6.2: Task sequences for the critical scenario, predicted execution times (in seconds), and proportion of time for interacting with the shared workspace versus doing individual work in the private workspace.

S#	Tasks	Time/s	Collaborative	Individual
S1	1) RESERVE 2) DRAW 3) RELEASE	3.6 + 3.6 + 3.6 = 10.8	$\frac{7.2\,\text{s}}{10.8\,\text{s}} = 67\,\%$	$\frac{3.6\,\text{s}}{10.8\,\text{s}} = 33\,\%$
S2	1) RESERVE 2) DRAW 3) DRAW 4) RELEASE	3.6 + 3.6 + 3.6 + 3.6 = 14.4	$\frac{7.2\,\text{s}}{14.4\,\text{s}} = 50\,\%$	$\frac{7.2\,\text{s}}{14.4\,\text{s}} = 50\,\%$

Step 4: Focus on User Goals. Given an appropriate cell in the shared workspace, each player carries out individual work by following one of two possible sequences of tasks, shown in Table 6.2. Sequence S1 corresponds to drawing a single connection between two points; sequence S2 applies to cases in which two connections can be drawn in the same cell.

Table 6.2 shows that in sequence S1 the majority of the time goes to collaborative tasks RESERVE and RELEASE, with 7.2 s or 67 % of the total predicted time; the actual connection drawing takes up only 33 %. It is therefore likely that the groupware designer admits that players will avoid such situation and instead prefer sequence S2, due to its lower collaboration overhead (50 %). However, S2 takes more time to execute, which means that other players will not be able to use the cell for a longer time span. This is discussed in the next step.

Step 5: Focus on Group Goals. In this step, task sequences S1 and S2 are evaluated in terms of productivity, opportunities, and restrictions. Table 6.3 shows that S2 is more productive than S1 because it takes 14.4 s to draw 2 line connections, thus the 8.3 gu/min, in contrast with the 5.5 gu/min of S1. Additionally, S2 also compares favourably with S1 in creating new opportunities for the other players: 20.8 gu/min versus 11.1 gu/min.

The logic behind the number of opportunities for each task sequence is illustrated in Figure 6.3. With sequence S1 only one vertical connection can be drawn by Sophie in cell 5, which, in the best case, opens two new opportunities to Charles since he will be able to draw two extra horizontal

Table 6.3: Productivity, opportunities, and restrictions for the two sequences of tasks that implement the critical scenario. Sequence `S2` is more productive and potentially creates more new opportunities for the other players, but it restricts access to the cell for a longer period of time.

S#	Productivity	Opportunities	Restrictions
`S1`	$\frac{1\,\text{gu}}{10.8\,\text{s}} = 5.5\,\text{gu/min}$	$\frac{2\,\text{gu}}{10.8\,\text{s}} = 11.1\,\text{gu/min}$	$1\,\text{gu} \times 10.8\,\text{s} = 0.18\,\text{gu min}$
`S2`	$\frac{2\,\text{gu}}{14.4\,\text{s}} = 8.3\,\text{gu/min}$	$\frac{5\,\text{gu}}{14.4\,\text{s}} = 20.8\,\text{gu/min}$	$1\,\text{gu} \times 14.4\,\text{s} = 0.24\,\text{gu min}$

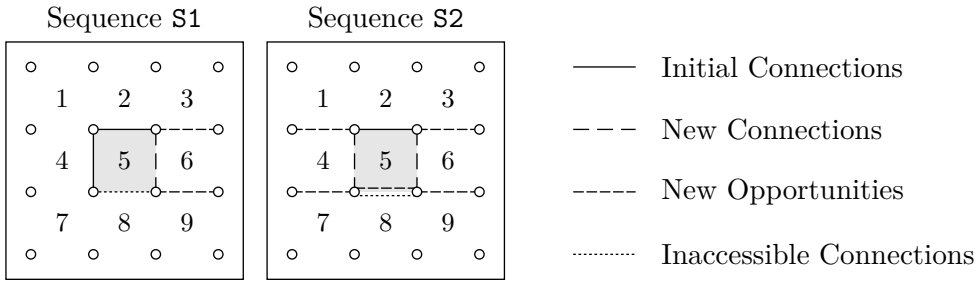


Figure 6.3: Opportunities created by each task sequence. Sequence `S1` creates up to two new opportunities for other players to draw connections. `S2` creates five opportunities at best, which suggests that it allows the group to progress faster.

connections at the top and bottom of cell 6. The missing bottom connection in cell 5 is *not* an opportunity because it was already available via the left vertical connection in cell 5. Actually, that bottom connection is inaccessible to the other players while Sophie is doing `S1`. In `S2` up to 5 opportunities can be created after the left and right vertical lines are drawn in cell 5.

The only dimension in which task sequence `S1` is preferable to `S2` is the restrictions to the work of the other players. Its lower 0.18 gu min versus 0.24 gu min of `S2` is only due to its lower execution time, 10.8 s versus 14.4 s, since the number of inaccessible goal units during the execution of the task sequence is the same in both cases: a single connection drawing (for instance, the bottom horizontal connection in cell 5 in Figure 6.3).

The data in Table 6.2 and Table 6.3 provide a basis for doing comprehensive comparisons of groupware usability, beyond task execution time, by highlighting the trade-offs between user and group performance associated with each critical scenario supported by the groupware interface.

Table 6.4: Collaborative tasks in the alternative design. Players now have the possibility of reserving or releasing multiple cells at the same time.

Task	Description/KLM Representation	Time/s
SELECT_1	1) Locate cell in workspace; 2) point mouse cursor to cell; and 3) press and release mouse button 1) M 2) P 3) BB	2.5
SELECT_N	1) Locate cell in workspace; 2) point mouse cursor to cell; 3) press mouse button; 4) locate second cell that defines the selection rectangle; 5) point mouse cursor to second cell; and 6) release mouse button 1) M 2) P 3) B 4) M 5) P 6) B	4.8
RESERVE_B	1) Press mouse button; 2) point mouse cursor to private workspace; and 3) release mouse button 1) B 2) P 3) B	1.3
RELEASE_B	1) Press mouse button; 2) point mouse cursor to shared workspace; and 3) release mouse button 1) B 2) P 3) B	1.3

6.3.2 Evaluating a Design Alternative

I now evaluate a design alternative for the collaborative game that features multiple cell reservations and releases as well as the display of awareness information while players *select* cells in the shared workspace. The motivation is twofold: a) the impact of collaborative overhead decreases with the number of connections that can be drawn consecutively in the private workspace; and b) selecting cells in the shared workspace is faster than reserving them, which means that awareness information will be more up-to-date.

The new features introduce changes in the *collaborative* tasks that define the critical scenario: two novel tasks are used for selecting single and multiple cells, SELECT_1 (a single click on a cell selects it) and SELECT_N (a click and drag movement for selecting consecutive cells); additionally, the reserves and releases, RESERVE_B and RELEASE_B, are now slightly simpler because players do not need to search for a cell or cells that they have just selected (cell selections always precede cell reservations or releases).

Table 6.4 shows the textual descriptions for the new tasks and also includes the sequences of KLM operators and predicted execution times, combining steps 2 and 3 of the evaluation method (step 1 was unchanged).

Table 6.5: New task sequences for the critical scenario. `S3` and `S4` in the alternative design are similar to `S1` and `S2` in the initial design. `S5` is entirely new, and allows multiple cells to be selected, reserved, worked, and finally released.

S#	Tasks	Time/s	Collaborative	Individual
`S3`	1) `SELECT_1` 2) `RESERVE_B` 3) `DRAW` 4) `SELECT_1` 5) `RELEASE_B`	$2.5 +$ $1.3 +$ $3.6 +$ $2.5 +$ $1.3 = 11.2$	$\frac{7.6\,\mathrm{s}}{11.2\,\mathrm{s}} = 68\,\%$	$\frac{3.6\,\mathrm{s}}{11.2\,\mathrm{s}} = 32\,\%$
`S4`	1) `SELECT_1` 2) `RESERVE_B` 3) `DRAW` 4) `DRAW` 5) `SELECT_1` 6) `RELEASE_B`	$2.5 +$ $1.3 +$ $3.6 +$ $3.6 +$ $2.5 +$ $1.3 = 14.8$	$\frac{7.6\,\mathrm{s}}{14.8\,\mathrm{s}} = 51\,\%$	$\frac{7.2\,\mathrm{s}}{14.8\,\mathrm{s}} = 49\,\%$
`S5`	1) `SELECT_N` 2) `RESERVE_B` 3) `DRAW` $\times\, n$ 4) `SELECT_N` 5) `RELEASE_B`	$4.8 +$ $1.3 +$ $3.6 \times n +$ $4.8 +$ $1.3 = \mathit{total}$	$12.2\,\mathrm{s}/\mathit{total}$ $n = 1 \rightarrow 77\,\%$ $n = 2 \rightarrow 63\,\%$ $n = 3 \rightarrow 53\,\%$ $n = 4 \rightarrow 46\,\%$	$(3.6\,\mathrm{s} \times n)/\mathit{total}$ $n = 1 \rightarrow 33\,\%$ $n = 2 \rightarrow 37\,\%$ $n = 3 \rightarrow 47\,\%$ $n = 4 \rightarrow 54\,\%$

The results in Table 6.4 show that `SELECT_1` requires less time to execute than the previous `RESERVE` task in Table 6.2 (2.5 s versus 3.6 s). This means that players should experience less time dealing with coordination conflicts, as awareness information (for instance, a letter identifying the player who is selecting the cell) is displayed in a more up-to-date fashion. On the other hand, the time to reserve a single cell slightly increases because now it takes a `SELECT_1` followed by a `RESERVE_B`, with a total of 3.8 s. This trade-off seems acceptable because the extra 0.2 s is much less than the time to recover from a reservation conflict, which would require a minimum of 1.2 s (an `M` operator) for recognising and resolving with it.

Table 6.5 shows the new task sequences for fulfilling the critical scenario as well as the new time proportions for interacting with the group and for doing individual work. This corresponds to step 4 of the evaluation method.

As expected, if players can *only* select single cells, they will probably prefer reserving those in which they can draw two connection lines using sequence `S4`, in detriment of `S3`. This is because in `S4` the overhead caused

Table 6.6: Productivity, opportunities, and restrictions for tasks in the alternative design. The results are ordered by restrictions (ascending), then by productivity (descending), and finally by opportunities (descending), to highlight tasks that are less obtrusive to the work of others.

S#	Productivity	Opportunities	Restrictions
S3	$\frac{1\,\text{gu}}{11.2\,\text{s}} = 5.4\,\text{gu/min}$	$\frac{2\,\text{gu}}{11.2\,\text{s}} = 10.7\,\text{gu/min}$	$1\,\text{gu} \times 11.2\,\text{s} = 0.2\,\text{gu min}$
S4	$\frac{2\,\text{gu}}{14.8\,\text{s}} = 8.1\,\text{gu/min}$	$\frac{5\,\text{gu}}{14.8\,\text{s}} = 20.3\,\text{gu/min}$	$1\,\text{gu} \times 14.8\,\text{s} = 0.3\,\text{gu min}$
S5(a)	$\frac{4\,\text{gu}}{26.6\,\text{s}} = 9.0\,\text{gu/min}$	$\frac{8\,\text{gu}}{26.6\,\text{s}} = 18.0\,\text{gu/min}$	$4\,\text{gu} \times 26.6\,\text{s} = 1.8\,\text{gu min}$
S5(b)	$\frac{6\,\text{gu}}{33.8\,\text{s}} = 10.6\,\text{gu/min}$	$\frac{10\,\text{gu}}{33.8\,\text{s}} = 17.8\,\text{gu/min}$	$6\,\text{gu} \times 33.8\,\text{s} = 3.4\,\text{gu min}$
S5(c)	$\frac{8\,\text{gu}}{41.0\,\text{s}} = 11.7\,\text{gu/min}$	$\frac{13\,\text{gu}}{41.0\,\text{s}} = 19.0\,\text{gu/min}$	$9\,\text{gu} \times 41.0\,\text{s} = 6.2\,\text{gu min}$
S5(e)	$\frac{9\,\text{gu}}{44.6\,\text{s}} = 12.1\,\text{gu/min}$	$\frac{12\,\text{gu}}{44.6\,\text{s}} = 16.1\,\text{gu/min}$	$10\,\text{gu} \times 44.6\,\text{s} = 7.4\,\text{gu min}$
S5(d)	$\frac{12\,\text{gu}}{55.4\,\text{s}} = 13.0\,\text{gu/min}$	$\frac{16\,\text{gu}}{55.4\,\text{s}} = 17.3\,\text{gu/min}$	$14\,\text{gu} \times 55.4\,\text{s} = 12.9\,\text{gu min}$
S5(f)	$\frac{12\,\text{gu}}{55.4\,\text{s}} = 13.0\,\text{gu/min}$	$\frac{16\,\text{gu}}{55.4\,\text{s}} = 17.3\,\text{gu/min}$	$16\,\text{gu} \times 55.4\,\text{s} = 14.8\,\text{gu min}$
S5(g)	$\frac{16\,\text{gu}}{69.8\,\text{s}} = 13.8\,\text{gu/min}$	$\frac{21\,\text{gu}}{69.8\,\text{s}} = 18.0\,\text{gu/min}$	$21\,\text{gu} \times 69.8\,\text{s} = 24.4\,\text{gu min}$

by collaborative tasks, 51 %, is lower than the 68 % in S3 (as had happened with S1 and S2 in the initial design, shown in Table 6.2). However, if players get a chance to reserve multiple cells at once, then they will likely use sequence S5 when at least four connections ($n \geq 4$) are drawable in those cells, because in these circumstances the impact of the collaborative tasks is 46 %, this being unmatched by sequences S3 and S4.

Table 6.6 shows the new group performance values afforded by the groupware interface in the alternative design, obtained by applying step 5 of the evaluation method to task sequences S3 and S4, as well as to several variants of sequence S5, illustrated in Figure 6.4.

The first rows in Table 6.6 contain sequences S3 and S4, which are less restrictive and offer good opportunities, albeit with lower productivity. In

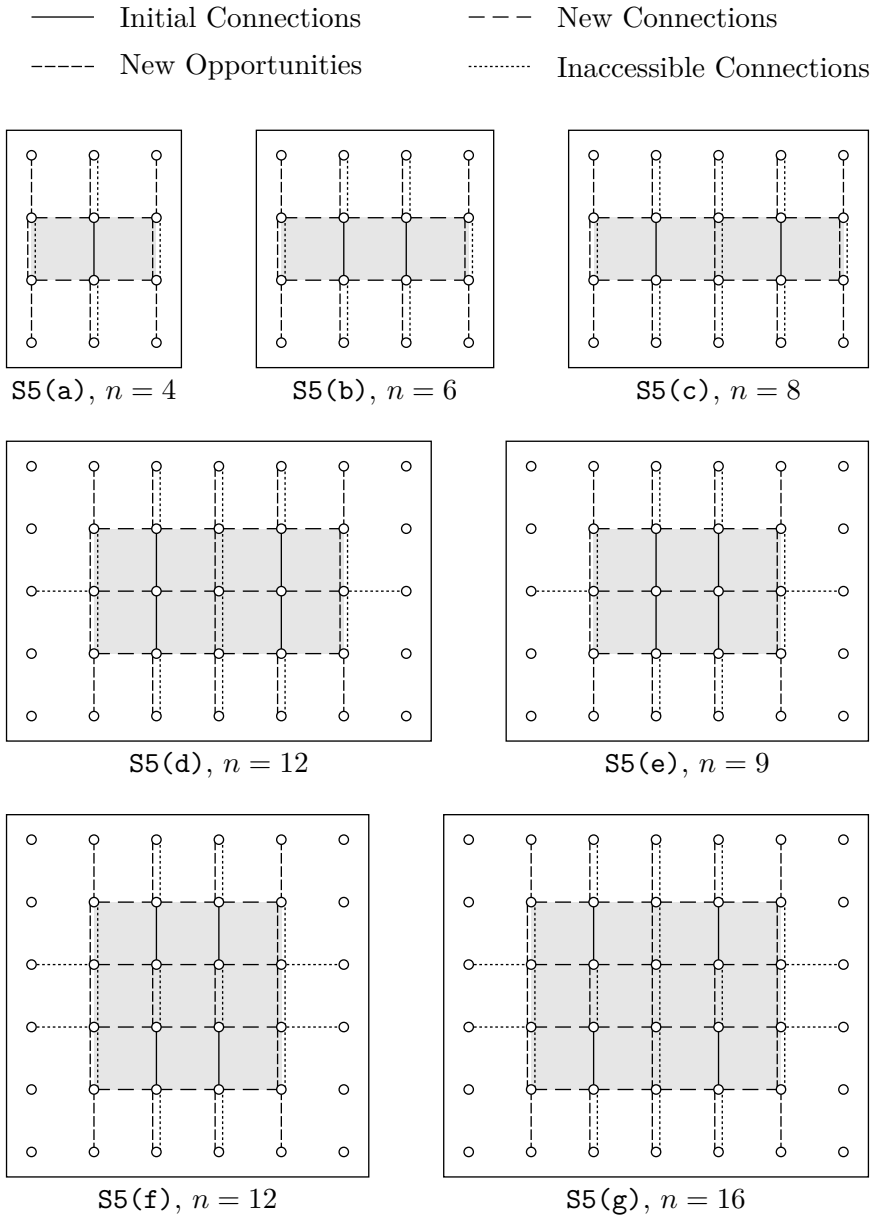


Figure 6.4: Variety of instances of task sequence S5. The n value corresponds to the number of cells that are reserved simultaneously. Greater values of n increase individual productivity by lowering collaborative overhead (see Table 6.5) but also the bigger the region in the game board that is unavailable for the other players.

the last rows are the more productive variants of S5, which are also the most restrictive and offer only normal opportunities to the other players. This arrangement of results facilitates the detection of sequences of tasks that have equal productivities and equal opportunities, but different restrictions. In such conditions group performance is better when users execute sequences of tasks that impose lower restrictions to the other members: for instance, S5(d) is better for the group than S5(f).

I end the evaluation of the design alternative by noting that the S5 variants in Figure 6.4 are ideal cases and that actual group performance afforded by the groupware interface depends upon the evolving state of the board. However, an exhaustive analysis of S5 variants is clearly unmanageable. By focusing on ideal cases of S5, a reasonable basis for making usability comparisons can be established.

6.3.3 Comparing Designs: The Big Picture

I now compare the two competing designs using the outcomes from the evaluation method. Figure 6.5 shows the impact of collaborative overhead in total predicted execution time for all task sequences that support the critical scenario. The entries in the figure are sorted by collaborative overhead to facilitate the detection of the task sequences that are more costly to perform in the shared workspace.

The data in Figure 6.5 show that sibling task sequences S3/S1 have similar collaborative overhead, and that the same happens with S4/S2. So far, it is difficult to tell which design to prefer. However, the variants of S5 have the best proportions of individual work in total predicted time. These results *seem* to indicate that the alternative design is preferable to the initial design, even more so because, intuitively, collaborative overhead has a negative effect in group performance.

However, this intuition may be wrong in some cases because it only reflects the user point of view. In fact, in the case of the collaborative game this intuition is actually *wrong.* I start by stating the following proposition: lower proportions of collaborative overhead for achieving the critical scenario lead to higher group performance towards the common goal. Now, consider the succession of S5 variants, with equal ordering in Figure 6.5 and Figure 6.6.

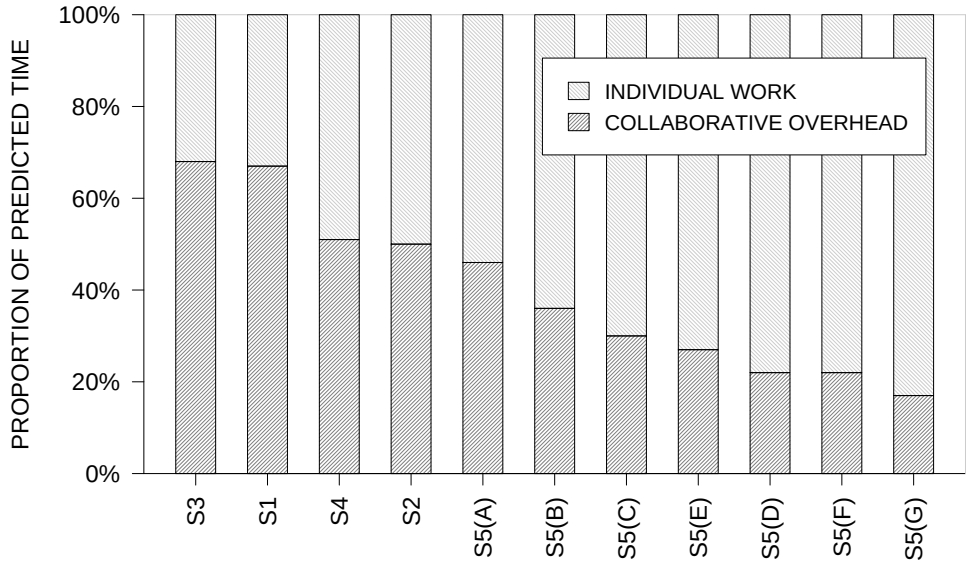


Figure 6.5: Collaborative overhead versus individual work for all task sequences that implement the critical scenario. `S1` and `S2` are from the initial design and the other sequences are from the alternative design.

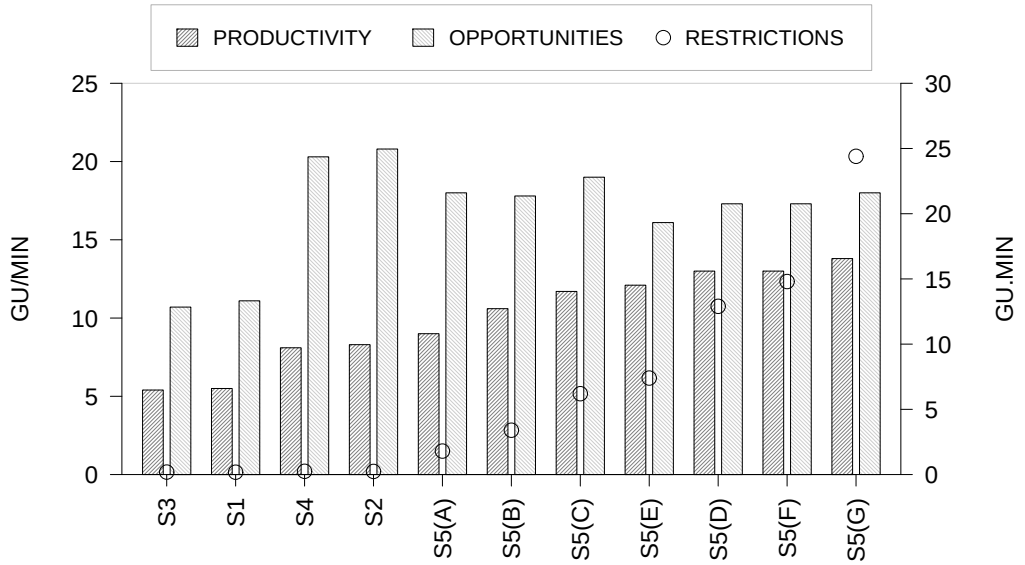


Figure 6.6: Comparison of productivity, opportunities, and restrictions. Sequences of tasks with greater productivity also impose much greater restrictions, and opportunities remain relatively constant.

Reading both plots in synchrony from left to right, the proportion of collaborative overhead steadily decreases (see Figure 6.5), productivity increases in a symmetrical way, opportunities remain relatively constant and restrictions increase at a much higher rate (see Figure 6.6). So, contrary to the proposition, the lower the proportion of collaborative overhead in the variants of `S5` the *slower* the group progresses towards the common goal because users will spend more time waiting to proceed.

Given this somewhat puzzling scenario the designer must *still* find an optimal equilibrium between user and group goals. Where this equilibrium could be is the subject of further work. At the moment the big picture is still getting clearer.

6.4 Discussion

The method presented in this chapter addresses a recurrent trade-off in the design of groupware systems for mixed-type collaboration: should priority be given to the users, so they can be more productive when working individually, or is group performance more important? What the method shows is that there is no easy answer, for two reasons.

Firstly, it would appear that by considering a more comprehensive view of groupware usability, comprising productivity, opportunities, and restrictions, which aims at capturing computer support for the intertwined nature of mixed-focus collaboration, the choice would become more clear. However, as I concluded at the end of the previous section, these three dimensions may very well confront the designer with yet another trade-off.

Secondly, the method concentrates on a very specific type of mixed-focus collaboration: one in which the group has a quantifiable goal to accomplish and in which critical scenarios executed by individual users offer measurable contributions to group progression towards the common goal. However, this is not frequently found in practise because group work often cannot be broken down into predetermined units. It is much more usual to find collaborative tasks in which both the common goal and the individual contributions are unmeasurable and even unknown beforehand, such as in decision making, planning, and idea generation.

Nevertheless, the example in this chapter shows that the method can be applied to evaluate the usability of a collaborative game, which suggests that video games may be a real-world application area.

At any rate, the three proposed dimensions of groupware usability raise the designer's awareness on the complexities of designing for users and for groups, and might eventually contribute to better decisions by pointing at three distinct directions, rather than simply to task execution time.

6.5 Summary

In this chapter, I presented a method for evaluating the usability of mixed-focus workspaces, a type of groupware in which users can work individually in private workspaces and interact with the group via a shared workspace, and showed how it can be applied in a collaborative game.

The method begins with the identification of critical scenarios of collaboration, whose task execution times are predicted using engineering models of human performance. Then, these execution times are combined with work measures, namely productivity, opportunities, and restrictions (see page 96), which aim at simultaneously capturing the intertwined nature of mixed-focus collaboration and the usually conflicting goals of users working as individuals or as elements of a group.

With this method, the groupware designer has a more comprehensive view of usability in mixed-focus workspaces, which may lead to better decisions.

Notes

A preliminary version of this method, with only three steps, was presented by my adviser at the tenth International Conference on Computer Supported Cooperative Work in Design (CSCWiD'06), held in Nanjing, China (Ferreira and Antunes, 2006a).

Later, I added two steps to the method, namely the characterisation of the group goal and the description of the critical scenarios, and updated the collaborative game evaluation. This research was published in a compilation of selected papers, *CSCW in Design III* (Ferreira and Antunes, 2007b).

Chapter 7

Drawing Attention to Cognitive Limitations

In the previous chapters, I described my contributions to formative evaluation of groupware usability using engineering models of human performance, with the purpose of providing instruments to make fine-grained optimisations without requiring real users or functioning prototypes. These models have shown to be representative of real users in numerous scenarios of human-computer interaction (see Section 3.3 on page 43), despite assuming users are tireless experts who never make mistakes.

In this chapter, I highlight the need for evaluating groupware usability with a focus on human information processing limitations, in particular regarding *information overload*, which, in broad terms, happens when the inflow of information exceeds our attentive capacity. I will present arguments for the increased likelihood of information overload happening during group work, which justifies an expansion of cognitive-level groupware evaluations beyond the application domain of methods grounded on engineering models.

7.1 Information Overload

Information overload is an important problem in our information-rich world: it is estimated that 23 exabytes (or 2.3×10^{19} bytes) of *new* data were produced in 2003, comprising paper, film, magnetic, and optical storage mediums,

as well as electronic flow mediums such as radio, television, telephony, and the Internet, the latter accounting for approximately 530 petabytes (or 5.3×10^{17} bytes) of new or updated content in Web sites, conversations via e-mail, and instant messaging (Lyman and Varian, 2003).

More recently, the size of the 'digital universe' was estimated to be 281 exabytes (or 2.8×10^{20} bytes) in 2007, comprising databases, e-mail, video conferences, instant messages, telephony, television, office applications, and other worldwide data sources (Gantz, 2008). Furthermore, about 35 % of these data originated in organisations, mostly from workers at their desks or on the road, corresponding to approximately 98 exabytes. In addition, the same report predicts that the amount of information produced in 2011, but not necessarily consumed or even stored, will be 1.8 zettabytes (or 1.8×10^{21} bytes), for an expected 60 % annual growth.

An even more recent study concluded that in 2008 an average American consumer was exposed to 34 gigabytes (or 3.4×10^{10} bytes) of data on an average *day*, emanating from sources such as newspapers, books, television, radio, telephony, movies, computer games, cameras, Internet communication, and so on, but excluding information at work (Bohn and Short, 2009).

Such immense quantity of information makes heavy demands on human processing capabilities, namely it creates a *scarcity of attention:* we simply cannot attend all information that surrounds us.

This condition has long been acknowledged within organisations and, in fact, four decades ago Simon (1971) had already warned that 'a wealth of information creates a poverty of attention' and had highlighted the need for efficient allocation of attention. Thus, organisations have been adopting countermeasures against information overload such as data filters and summarisers (Eppler and Mengis, 2004, Table 5) and yet the problem persists in diverse scenarios, for example, information search, decision making, investment analysis, and many others (Eppler and Mengis, 2004, Table 1).

7.1.1 Complexities of Group Work

Following up on the idea that human attention must be preserved, I argue that during group work users are more exposed to information overload because collaboration is more demanding compared to individual work:

- People working on a group have to deal with *larger amounts of information* due to the extra communication effort needed to coordinate work; this effort usually grows exponentially with the size of the group and may quickly outweigh the benefit of admitting new people, a situation that has been captured in Brooks's Law: adding manpower to a late project makes it later (Brooks, 1995, Ch. 2);
- Besides the increase in the quantity of information, users doing group work have to attend to *multiple information sources*, such as messages from colleagues and various displays with different perspectives of group activity (see Section 4.3.2 on page 63); this multiplicity of sources is known to be more important than the rate of information presentation in degrading user performance, as people tend to sample fewer sources when under stress (Sanders and McCormick, 1992, pp. 69–76);
- Group members have to explicitly manage the trade-offs between doing individual work and attending to the group, including handling interruptions from colleagues; this *work fragmentation* is estimated to occur in 57 % of the tasks of information workers, and may become detrimental due to the stress in keeping up with multiple task states and the extra cognitive cost in resuming work (Mark et al., 2005).

In these circumstances, human attention is more likely exercised to the point that relevant information is discarded or quickly forgotten, stress and confusion build up, errors become more frequent, among other symptoms compiled in Eppler and Mengis (2004, Table 4). In fact, there is a growing body of evidence showing that memory failures regarding tasks yet to be performed are becoming a significant problem for information workers, leading people to devise countermeasures such as emailing reminders to themselves (Czerwinski et al., 2004).

Naturally, the net influence of these symptoms may reduce the group's ability to build and maintain a shared awareness of the current situation and may also penalise the understanding of how the group plans its activities. Ultimately, information overload may also restrict the development of organisational memory, potentially leading to cases in which no one knows where documents are or how to conduct certain work processes (Khoshafian and Buckiewicz, 1995, pp. 40–43).

7.1.2 Influences from Groupware Research

To complicate matters concerning information overload during group work, a popular and influential trend in groupware research has been to design systems that provide *ever greater* awareness information about the presence and actions performed by users on a group. This trend has been driven by the need to counter the relative inefficiency and clumsiness of collaboration via computer-controlled communication channels compared to face-to-face interaction (Gutwin and Greenberg, 2002), and has been pointing towards the discovery of a comprehensive set of group awareness devices, an intention perhaps made more explicit in Raikundalia and Zhang (2005).

It is thus unsurprising that a significant research effort has been devoted to the development of toolkits designed to facilitate the rapid prototyping and testing of groupware tools and, more importantly here, to provide off-the-shelf group awareness devices that can be quickly arranged and reused in multiple ways, and even extended with additional devices, as needed. Some recent and early toolkits and their motivations include:

- MAUI, or Multi-User Awareness UI Toolkit, which contains groupware-specific devices and adaptations of existing single-user devices to the group context, and aims at 'simplifying the construction and testing of rich groupware interfaces' (Hill and Gutwin, 2004);
- GroupKit, which provides group session management and a set of graphical user interface elements for building conferencing applications, and emphasises the 'belief that programming groupware should be only slightly harder than building functionally similar single-user systems' (Roseman and Greenberg, 1996);
- Rendezvous, which offers replicated views and graphical interface components to 'simplify the construction of multi-user applications for real-time collaboration' (Hill et al., 1994);
- MMConf, which offers shared workspace management for teleconferencing, and aims at supporting 'highly graphical, highly interactive applications' (Crowley et al., 1990);
- LIZA, which provides shared windows and remote cursors in an 'extensible system for exploring multi-user interfaces' (Gibbs, 1989).

In other words, a major design problem being addressed by groupware researchers especially through the use of groupware toolkits is that of information scarcity: more group awareness devices must be discovered and made readily available to answer questions such as who is collaborating, where is activity being carried out, and what is going on; and in this way computer-mediated collaboration will become more natural and efficient.

However, this trend seems to ignore that *more may be less* if information overload occurs, as this possibility is rarely, if ever, mentioned, despite early evidence showing precisely the contrary (Stefik et al., 1986). In addition, this approach may actually be going against its intended motivation, as collaboration under information overload is likely to become even less efficient and more unnatural.

7.1.3 Designing for Attention Scarcity

Given this situation, I argue that the features and usability of groupware systems influence the likelihood of information overload arising during collaboration, and consider two opposing effects in human attention:

- A poorly designed system, for example, with many simultaneous information sources and too much information overall, may impose stress to the users' capacity to attend to the information flows; and
- More importantly in the context of this dissertation, users may benefit if the groupware system is designed to support the *goals* and compensate the *limitations* of human attention.

Thus, instead of following the dominant trend of presenting more information to the users through an array of group awareness devices, I propose an approach that aims at preserving the users' attention via novel *attentive devices* adapted to scenarios of information overload. Figure 7.1 shows the contrast between designing for information versus attention scarcity.

As a matter of fact, the use of computers to support human attention is gaining momentum, as evidenced by recent research on Attentive User Interfaces (AUI) (Vertegaal, 2003; Roda and Thomas, 2006). However, because of its roots in the Human-Computer Interaction (HCI) field, the focus of AUI has been mostly on single-user systems, which means that evaluating the usability of attentive groupware systems is largely an unexplored area.

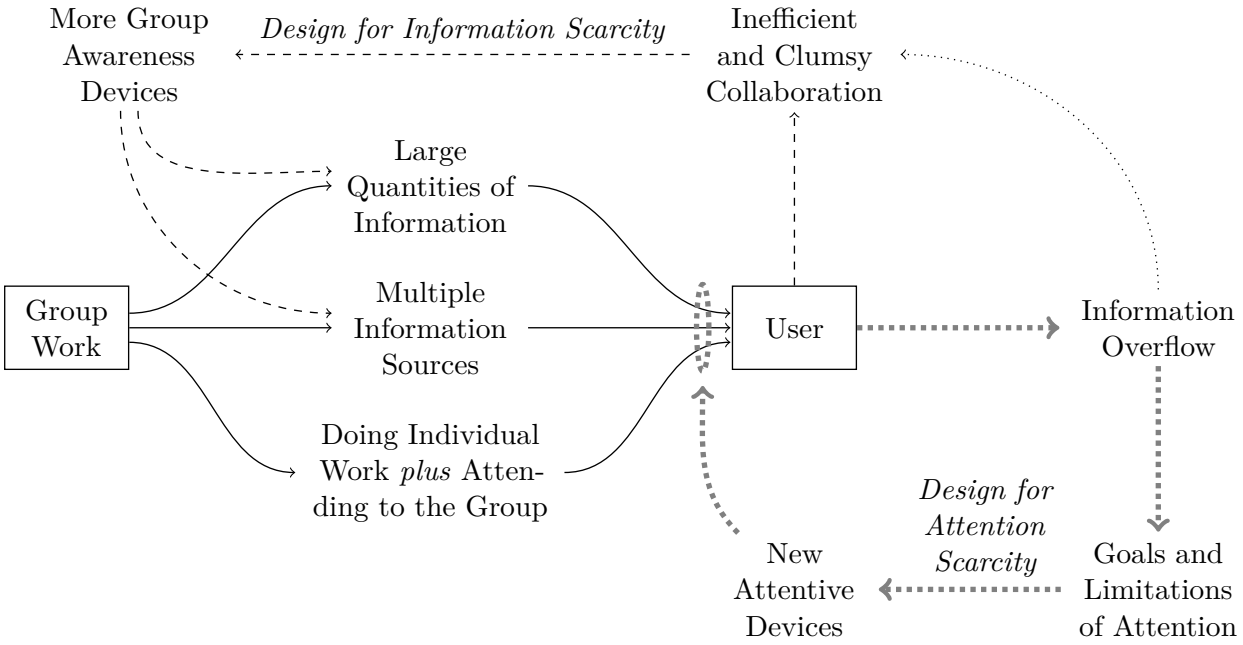


Figure 7.1: Information overflow during group work. Users working on a group are subject to large quantities of information, have to attend to multiple information sources, and must manage the alternation between doing individual work and keeping up with the group. These circumstances increase the likelihood of information overload occurring, which highlights the need for designing groupware systems for attention scarcity, through novel attentive devices that manage the information reaching each user, an approach represented by the ⋯→ cycle. However, an influential trend in groupware research is driven by the need for more group awareness information (the - - → cycle) as a way to handle inefficient and clumsy collaboration, which may actually exacerbate information overflow. In addition, information overflow may make collaboration even less efficient and more unnatural (the ⋯→ link).

In the next sections, I review the goals and limitations of human attention, and then provide an overview of existing AUI technologies designed to support attention in multi-user and single-user computer systems.

7.2 Human Attention

Human attention is associated with the selection of relevant information and attenuation or discard of non-relevant data. It is a process that optimises the use of our limited cognitive resources so that we can perceive and act accurately and quickly (Anderson, 2005, Ch. 3; Sternberg, 2003, Ch. 3; Eysenck and Keane, 2000, Ch. 5).

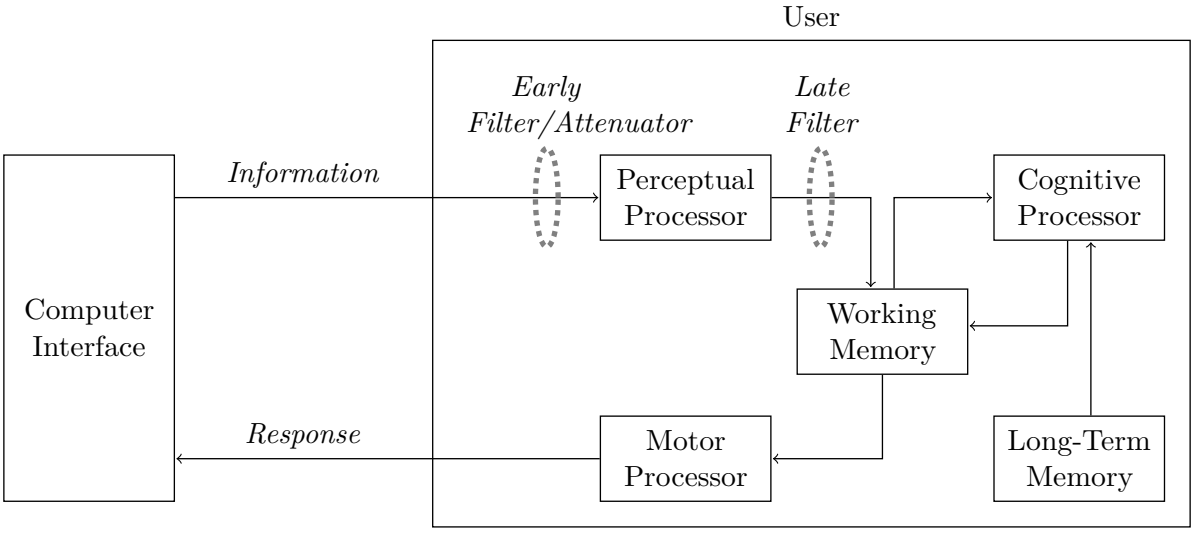


Figure 7.2: Role of attention in the Model Human Processor, adapted from Figure 3.1 on page 34 and based upon theories illustrated in Sternberg (2003, Figs. 3.5 and 3.6), Anderson (2005, Fig. 3.5), and Eysenck and Keane (2000, Fig. 5.2). Theoretically, attention is a filtering/attenuation mechanism that reduces the amount of information reaching the working memory to preserve resources in the cognitive processor and facilitate the selection of a response.

Depending on the theory, attention is thought to act as an early or late information selection mechanism. According to Broadbent's theory, stated in 1958, attention is an early filter that rejects non-relevant information before it reaches the perceptual processor. Another theory, by Treisman in 1964, argues that attention is an early attenuator of information importance that does not completely block any particular piece of information.

These filtering and attenuation mechanisms can be represented in the Model Human Processor to the left of the perceptual processor, intercepting the flow of information generated by the computer interface (see Figure 7.2).

A third theory, by Deutsch and Deutsch in 1963, posits that attention is a late filtering mechanism that acts after some form of sensory analysis has taken place in the perceptual processor, shown in Figure 7.2 as a late filter.[1]

Despite the different theories, a common assumption is that attention is a *bottleneck* through which only the most relevant information passes on to the working memory, so that less interpretation effort is imposed on the cognitive processor, thus allowing for faster response times.

1 More information about these three theories can be found in Anderson (2005, pp. 75–79), Sternberg (2003, pp. 92–95), and Eysenck and Keane (2000, pp. 121–123).

The goals and limitations of the attentional bottleneck have been of interest for psychologists over the decades: the *goals* are usually defined in terms of accuracy and speed responding, both contributing to decrease task execution times; the *limitations* occur when we take longer to respond to a stimulus or when we fail to detect changes in the information before us, thus penalising task performance, and are associated with phenomena such as the psychological refractory period, attention blink, and change blindness.

7.2.1 Goals of Attention

Two of the main goals of attention are accuracy, to perceive specific objects or to execute particular tasks, and speed responding, to perceive objects or execute tasks after the presentation of a predictive cue (LaBerge, 1999).

Accuracy occurs when we successfully remove or attenuate the influence of extraneous and confusing information (LaBerge, 1999, pp. 45–46), as happens when we manage to keep track of a conversation in a crowded room—the 'cocktail party' phenomenon (Anderson, 2005, pp. 91–92; Eysenck and Keane, 2000, p. 121). Curiously, people with low working memory seem more likely to suddenly realise their name was spoken elsewhere, which suggests they may have more difficulty blocking out distractions (Conway et al., 2001).

Regarding groupware support for attentional accuracy, one of the few examples is GAZE-2, a video conferencing system that automatically regulates the sound volume of overlapping conversations according to the direction each user is gazing at (Vertegaal et al., 2003). I provide a more detailed description of this system in the next section.

Speed responding manifests itself when we are able to respond faster to an anticipated event following the presentation of a predictive cue, and almost always involves an expectation of an upcoming time to initiate the response (LaBerge, 1999, pp. 45–47).

For instance, consider the traffic lights, which, in some countries, turn red, then yellow, and finally green instead of switching directly from red to green. The extra yellow light is a predictive cue that prepares the driver to react more quickly to the green light. Another example, from an experiment concerning visual attention, shows that the response to expected signals is about 11 % faster than to unanticipated signals (Anderson, 2005, p. 80).

Groupware support for speed responding is almost ubiquitous given that most Internet instant messengers have typing indicators turned on by default (Saunders, 2004). As the system detects user activity whose final outcome may be pertinent to another user, such as when someone is typing a reply, it generates a predictive cue, for instance, a swinging pencil next to the photo of the user who is typing, to prepare the recipient for the imminent arrival of a new message and thus make the conversation more efficient.

7.2.2 Limitations of Attention

As we all know, human attention is limited; distraction is part of everyday life and we find it natural that sometimes we react slowly to external stimuli or even miss them altogether. Psychologists have conducted numerous experiments to determine the circumstances in which our attentional resources restrict performance, of which I highlight three phenomena that are well documented in the literature and that are related to the processing of dynamic flows of information, typical of fast-paced group work settings.

The first phenomenon is the *psychological refractory period*, or PRP, and corresponds to a slowdown in the response speed to the second of two stimuli presented in rapid succession, an effect that is still observable after tremendous training (Eysenck and Keane, 2000, pp. 137–138).

Experiments with the PRP are characterised by two distinct tasks which must be executed as fast as possible in response to two different types of stimuli; for example, the stimuli may be a tone and a light and the tasks may be pressing a key and saying a word, respectively (Wickens and McCarley, 2008, pp. 10–13). Interestingly, response time to the first stimulus is generally constant, which is compatible with the view of attention as an information processing bottleneck, in that only one stimulus is attended to at a time. In other words, if the second stimulus is presented before the first one is completely processed, a delay occurs because the second task is put on hold.

Another phenomenon is the *attentional blink*, or AB, which happens when we have difficulty detecting the second of two targets presented in rapid succession amid a series of distractors (Shapiro et al., 1997).

Despite also exposing people to fast-paced information, attentional blink experiments differ from PRP research in a number of ways (based upon

Table 7.1: Attentional phenomena in fast-paced information flows. The psychological refractory period (PRP) and the attentional blink (AB) occur in distinct circumstances and are assessed differently but both reduce performance in the second of two tasks, where attention to a second stimulus is required.

Phenomenon	**Stimuli**	**Distractors**	**Tasks**	**Performance**
PRP	Distinct	No	Distinct	Response time
AB	Similar	Yes	Similar	Detection rate

Wong, 2002): firstly, the stimuli are all of the same type, say two target letters; secondly, the stimuli are surrounded by distractors, such as other letters; thirdly, the tasks are very similar, for example to detect and identify the first and second target letters; and fourthly, task performance is measured in target detection rate rather than response time (see Table 7.1).

Nevertheless, as also happens in the psychological refractory period phenomenon, performance under the attentional blink is generally constant for the first stimulus, and gradually improves for the second stimulus as the time between consecutive stimuli[2] increases, up to a natural limit (for instance, compare Marois and Ivanoff, 2005, Figs. 1b and 3a).

This evidence suggests that groupware systems should compensate for the PRP and AB phenomena, for instance, by transforming unpredictable bursts of information generated in parallel by the users on a group into more steady flows of information, perhaps delivered in batches. To the best of my knowledge, this manipulation has not been tried.

The third attentional phenomenon is *change blindness*, or CB, which occurs when we fail to notice that something is different from what it was in the environment around us, even when the changes are large, repeatedly made, and anticipated (Simons and Rensink, 2005; Rensink, 2002, p. 246), and even without user fatigue or stress (Durlach, 2004).

Numerous scenarios with change blindness have been documented in the literature, such as drivers not noticing that the car ahead now has its brake lights on and pilots failing to notice changes in the flight mode (Wickens and McCarley, 2008, pp. 22–23). Surprisingly, we may not even detect that the

2 The time between consecutive stimuli is usually called stimulus onset asynchrony, or SOA (Marois and Ivanoff, 2005; Wong, 2002; Shapiro et al., 1997) and also interstimulus interval, or ISI (Wickens and McCarley, 2008, p. 11).

person we were speaking to was swapped by someone new during a casual face-to-face conversation (Anderson, 2005, Fig. 2.30).

The evidence on this phenomenon suggests that unless we are focusing our attention on a changing object, we tend to miss the modifications, especially if they fit into the context (Anderson, 2005, pp. 69–70). If we do want to check if anything has changed, then we have to engage in a very slow process of scanning the full picture in front of us, one object at a time. This happens because, although we can attend to four or five objects simultaneously, we can only detect one change at a time (Simons and Rensink, 2005).

The effect of change blindness in users doing group work is pertinent, since groups share a common context and the existence of several users contributing to the same goal stimulates scenarios in which multiple changes can occur simultaneously. This creates the conditions for people not noticing changes, which may affect the users' ability to catch up with the group and also likely damages reciprocal task interdependence.

To compensate for change blindness, groupware systems should highlight or bundle the changes so that they occur less frequently and are easier to notice. One example of both these techniques is the ubiquitous use of a bold face to highlight the arrival of a batch of new messages in e-mail in-boxes.

7.3 Attentive User Interfaces

In contrast with the previous overview of the goals and limitations of human attention, which is very much focused on the internal processes and behaviour of the human being, research on Attentive User Interfaces (AUI) is mostly concerned with providing technology to support attention.

During the late 1990s several researchers from the Human-Computer Interaction (HCI) field became interested in AUI and since then this area is gaining momentum, as evidenced by special issues in the Communications of the ACM (Vertegaal, 2003) and in Computers in Human Behavior (Roda and Thomas, 2006), and by specific conference sessions.[3]

3 The Human Factors and Computing Systems conference (`http://www.sigchi.org/conferences/chi`) organised sessions on 'don't interrupt me' in 2007, and 'attention and interruption' and 'task and attention' in 2008.

A prime motivation for AUI is the recognition that as the needs for information and communication rise so do the costs of not paying attention and being interrupted. So, instead of assuming the user is always focused on the entire computer screen, AUI negotiate, rather than impose, the user's attention by prioritising information presentation. To this end, researchers are enhancing input/output devices so that the user remains focused on a primary task without getting too much distracted by secondary tasks.

However, most research on AUI is directed towards single-user systems, whereas *multi-user* attentive interfaces are situated in video conferencing, which means that there is ample room for studying the application of AUI on other types of groupware systems.

7.3.1 Attentive Devices in Multi-User Systems

Research on attentive devices for groupware systems is mainly focused on the use of eye-trackers to facilitate the detection of who is talking to whom in remote meetings. The first such system was GAZE and it worked by showing photos of up to three users on the computer display, which could be rotated by intervention of eye-trackers placed in front of each user to reinforce the impression that some users are looking at the current speaker (Vertegaal, 1999). Afterwards, when the third user stops talking and looks at the photo of another user, that user knows s/he was given floor control and may begin to speak. In this way, the group turn taking process is more natural and requires less interruptions to determine who will speak next.

In GAZE-2, the photos of the users are replaced by live videos and also multiple simultaneous conversations are allowed (Vertegaal et al., 2003). In this system, the face of each user is captured by three video cameras from slightly different angles and an automated camera director chooses the best video stream to send to each one of the other users based upon the amount of parallax error as determined by an eye-tracker. As in GAZE, the virtual representation of each user is rotated to reflect his or her focus of attention, which typically corresponds to the current speaker.

Another feature of GAZE-2 is the automatic filtering of voices when multiple conversations are being held at the same time. Depending upon the user in focus, so is the respective audio stream amplified and the other

streams attenuated (but not eliminated). If the focus of interest suddenly changes, as sensed by the eye-tracker, the audio is again adjusted.

The concepts in GAZE-2 were further explored in eyeView, a groupware system that supports large remote meetings by manipulating the size of video windows and the voice volumes of each user on the group as a function of the current focus of attention (Jenkin et al., 2005).

7.3.2 Attentive Devices in Single-User Systems

In contrast with groupware systems that mainly rely on eye-trackers to augment human attention, a variety of input/output devices has been tested on attentive interfaces for single-user applications, including:

- Sensors that detect the user's focus of attention based upon eye-gaze and body orientation;
- Physiological sensors that assess the user's mental workload by measuring heart rate variability and pupil dilatation;
- Sensor-based statistical models that determine adequate moments to interrupt and communicate with the user; and
- Displays that present information at various levels of detail, depending upon the user's current focus of attention.

The following is a description of several applications of these input/output devices that I compiled from a survey of the literature.

Eye-Gaze and Body Orientation Sensors

Regarding the use of eye-trackers to support human attention, applications include magnifying the graphical window on which the user is currently focused, controlling a robotic directional microphone coupled to a video camera to overhear a specific conversation taking place in a remote room, and detecting eye contact to automatically choose which electronic appliance should obey to voice commands (Vertegaal et al., 2006, Figs. 7, 11, and 3).

In addition, Zhai (2003) used eye-gaze to point a cursor on the screen with minimal manual intervention, and Hyrskykari et al. (2000) monitored the user's gaze path during the reading of a foreign book to provide automatic translation when s/he pauses at a difficult word or re-reads a sentence.

Body orientation sensors are less precise than eye-trackers in depicting the user's focus of attention. Nonetheless, they have been tested in an office environment to regulate the transparency of cubicle walls—opaque when the user does not wish to attend requests from others—and to control noise cancellation in headphones (Vertegaal et al., 2006, Figs. 9 and 10).

Physiological Sensors

Concerning physiological sensors, these are used in AUI to assess mental workload, which is considered a surrogate of attentional capacity: the greater the workload, the lower the available attentional resources, for example, to handle unexpected emergencies (Wickens and McCarley, 2008, p. 4).

Chen and Vertegaal (2004) used heart rate variability to distinguish between at rest, moving, thinking, and busy states, and installed a regulator of notifications on a mobile phone that, for instance, minimises disruptions during a face-to-face conversation by automatically activating the phone's silent mode and setting the instant messenger status to busy. In the late 1990s, Rowe et al. (1998) had already suggested that heart rate variability could be used to assess conditions of *excessive* mental effort, that is, with information overload, in the monitoring of increasingly complex displays.

More recently, Bailey and Iqbal (2008) proposed using changes in pupil dilatation as an indicator of mental workload and combined this measure with GOMS models (see Section 3.2.2 on page 38) of route planning, document editing, and e-mail classification tasks. They realised that the workload depends on the type of task, that it varies during the execution of any of these tasks, and, most importantly, that it decreases at sub-task boundaries. Thus, they predict that an adequate moment to interrupt the user is between the completion of a sub-task and the beginning of the next.

Sensor-Based Statistical Models

Another approach to detect the best time to interrupt the user is to employ statistical models that continuously estimate and balance the value of information with the cost of interrupting, using sensors that capture work patterns, ambient noise, body posture, and also data from selected software applications, such as appointments in the personal calendar (Horvitz et al.,

2003). The initial model was hand-made by asking users to annotate video recordings with a description of their state of interruptibility at that moment and by relating this information with the data that had been recorded by the sensors, as explained in Horvitz and Apacible (2003).

Sensor-based statistical models were also used in Fogarty et al. (2005a) but with the following differences: firstly, the self-assessments of interruptibility were made intermittently while the users were working in their offices, instead of retrospectively in a separate video analysis session; secondly, the sensors recorded speech, writing, the number of people in the office, and whether the door was opened or closed; and thirdly, the sensors were simulated via the manual coding of the recorded video feeds. One of the results of this study is that statistical models can estimate human interruptibility as well as people do concerning high-level, social engagement, situations.

Interestingly, the same authors also conducted an experiment with low-level software sensors embedded in an integrated development environment that captured a myriad of events related to coding, navigating, debugging, among other activities that occur during a typical computer programming session (Fogarty et al., 2005b). To assert the state of user interruptibility, a secondary task was introduced, namely to do a mental multiplication, which could only be completed when the user clicked on a flashing notification button; the time between the notification appearing on the screen and the corresponding button press was then classified by humans in terms of interruptible, engaged, and deeply engaged. Results for this experiment were less conclusive, despite the favourable comparison against other systems.

Attentive Displays

Finally, the last attentive device that I refer to is a display that decreases the level of detail in the areas surrounding the user's visual focus of attention to reduce the viewer's mental workload (Baudisch et al., 2003).

7.4 Discussion

Confronting existing Attentive User Interfaces (AUI) with groupware usability evaluation and the information overload problem, I found no evidence

that such a research has been conducted before. In fact, most evaluation studies are directed towards the technological devices *per se* and do not consider the outcomes of using the devices in work settings.

For example, to the best of my knowledge, the GAZE-2 groupware system was evaluated through a user questionnaire that measured the users' self-perception of eye-contact and distraction, as well as changes in colour and brightness during camera shifts (Vertegaal et al., 2003), but no attempt was made to determine if GAZE-2 introduced any benefits to group work.[4]

Actually, a similar concern was expressed in Fogarty and Hudson (2007), regarding the trend to focus the evaluation of sensor-based statistical models of interruptibility on technical innovations and on analyses of model reliability, in detriment of determining the impact of these models on how people interact with computers. They proposed a toolkit to simplify the deployment of such models in applications, arguing that this will encourage researchers to look in new directions, despite the inevitable errors made by statistical models.

7.4.1 Evaluation of Attentive User Interfaces

Some studies *do* address the evaluation of AUI techniques in terms of task execution and user performance, most of them, if not all, inspired by experimental HCI research on *dual-task* performance, which, in turn, has been driven by the proliferation of computer systems and applications that proactively send notifications to users, for instance, to signal the arrival of new e-mail (McCrickard and Chewar, 2003).

The purpose of a typical dual-task experiment is to evaluate the effects on the execution of a primary task caused by the user being interrupted with requests to attend to secondary tasks. Table 7.2 shows a list of HCI studies concerning dual-task experiments, for which the earliest reference I could get access to dates back to the late 1980s (Field, 1987).[5]

4 In contrast, the MultiView system, which shares GAZE-2's goal of aiding floor control but does not consider user attention, has been evaluated for group trust formation and no difference was found compared to face-to-face collaboration (Nguyen and Canny, 2007).

5 Bailey and Konstan (2006) mention an even earlier study with calculator-based tasks, whose reference is: J. G. Kreifeldt and M. E. McCarthy. Interruption as a test of the user-computer interface. In *MC'81: Proceedings of the seventeenth annual conference on Manual control*, pages 655–667, Pasadena, CA, USA, 1981. Jet Propulsion Laboratory.

Table 7.2: Types of tasks in experimental HCI interruption research, ordered by year. The studies in this table assume that one user performs a primary task while being interrupted one or more times by a request to execute a secondary task. The task names have been adapted for consistency.

Study	Primary Task	Secondary Task	Interruption Rate
Iqbal and Bailey (2008)	Programming Diagram editing	Read hints Read news (other tasks)*	0.13 per minute (over 2 hours)
Gluck et al. (2007)	Memory game	Read hints	0.92 per minute (over 17 minutes)
Bailey and Konstan (2006)	Mental arithmetic Word counting Image comprehension Reading comprehension Form registration Word selection	News title decision	†
Iqbal and Bailey (2006)	Video editing Route planning Text editing Collage generation Form design	Stock decision	0.36 per minute (over 5–6 minutes)
Fogarty et al. (2005b)	Programming	Mental arithmetic	0.33 per minute (over 70 minutes)
Iqbal and Bailey (2005)	Route planning Text editing E-mail classification	News title decision	†
Adamczyk and Bailey (2004)	Text editing Video description Web searching	News title decision	†
McFarlane (2002)	Action game	Match symbols	17.8 per minute (over 4.5 minutes)
Cutrell et al. (2001)	List evaluation	Mental arithmetic	†
Czerwinski et al. (2000)	Drawing Spreadsheet editing Text editing	Stock decision	†
Field (1987)	Database search	Numeric sequence Manual search	†

*Users were allowed to check e-mail and read news during the execution of the primary task.

†The user was interrupted one or two times throughout the primary task rather than recurrently.

In this dual-task paradigm, it could well be that users are exposed to information overload, especially if the primary and secondary tasks involve large amounts of information and the user is interrupted often.

However, Table 7.2 shows that in most studies the secondary task exposes users to small amounts of information (for instance, two numbers that must be added) and that the interruption rate is low or even not applicable because in about half of the experiments the user was interrupted only once or twice over the duration of the primary task, instead of recurrently.

The exceptions to these considerations are the studies by McFarlane (2002) and, to a lesser extent, Gluck et al. (2007), in which the user was bombarded with interruptions while s/he was playing a simple, repetitive, computer game.

Even though there is little evidence of information overload occurring during the work tasks in Table 7.2, it is accepted that its likelihood increases if the user is frequently interrupted (Speier et al., 1999). Moreover, some of the studies from the same table show that as little as *one* or *two* interruptions to a primary task can have negative consequences:

- Bailey and Konstan (2006) reported that the time on the primary task increased up to 27 %, with an average of 13 %;
- Iqbal and Bailey (2005) showed that users felt about 89 % more annoyed when exposed to interruptions and that the computer was roughly 45 % less respectful of their work; and
- Adamczyk and Bailey (2004) obtained similar results, with about 97 % more annoyance and 33 % less respect due to interruptions, plus around 92 % more frustration and 27 % increased mental effort.

It was from this kind of evidence—which had already been advanced in previous studies such as Cutrell et al. (2001), Czerwinski et al. (2000), and Field (1987)—that a corpus of research began to emerge with the purpose of designing and evaluating AUI so that effective techniques could be found to reduce the cost of interrupting the user.

One technique that has been capturing the interest of researchers in recent years is to postpone interruptions until the user reaches sub-task boundaries within the primary task, where it has been posited that mental workload is lowest, thus leaving more attentional resources available for the

Table 7.3: Effects of deferring interruptions to task boundaries, relative to the condition in which interruptions occurred at random moments. This AUI technique has been successfully applied to programming, text and diagram editing, web searching, and other tasks shown in Table 7.2.

Study	Errors	Reaction Time	Resumption Time	Annoyance	Frustration	Anxiety	Time Pressure	Respect
Iqbal and Bailey (2008)		−25 %	*		−13 %			
Bailey and Konstan (2006)	−50 %			−36 %		−46 %		
Iqbal and Bailey (2005)			−64 %	−28 %				+39 %
Adamczyk and Bailey (2004)				−30 %	−17 %		−27 %	+27 %

*Measured, but no effect was found.

user to handle secondary tasks (Bailey and Iqbal, 2008). In other words, rather than being immediately interrupted at random moments relative to the primary task, the AUI defers the delivery of, say, an e-mail notification, until the user finishes the current sub-task or the main task itself.

Table 7.3 shows that this so-called *defer-to-boundary* (DTB) technique induces positive effects on users' performance and affective state: the number of errors while on the primary task was reduced by half; the reaction time to attend to the secondary task and the subsequent resumption time decreased; users felt less annoyed, less frustrated, less anxious, and less pressed by time; and the computer was found more respectful of the users' work.

To complete this picture of AUI evaluation, I add that Bailey and Konstan (2006) also contrasted the time on both the primary and secondary tasks with immediate versus deferred interruptions, and found no difference in the time users spent to complete the primary task, whereas deferring interruptions marginally *increased* the time on the secondary task.

7.4.2 Opportunity for Attentive Groupware Research

The results from the previous studies suggest that one way to mitigate information overload during computer-based tasks is to use an Attentive User Interface that defers interruptions to task boundaries, despite the existence of some contradictory data:

- On the one hand, typical consequences of information overload, such as a greater propensity for errors, and the buildup of stress and confusion (Eppler and Mengis, 2004, Table 4), seem to be favourably addressed by the DTB technique, as shown in Table 7.3;

- On the other hand, the extra time on the primary task due to interruptions at random moments (see page 128) was not counterbalanced at all by the DTB technique, and to make matters worse, the time on the secondary task increased (Bailey and Konstan, 2006).

I note that the theoretical basis for the DTB technique is plausible: it is logical to interrupt the user when mental workload is lowest (as long as immediateness is not critical) and it is reasonable to assume that task switching is less costly when the user has just completed an activity. Moreover, there seems to be more evidence in favour of deferring interruptions to task boundaries than against it, and I point out that the effects on primary and secondary task performance have only begun to be researched.

I view this situation as an *opportunity* to expand the body of knowledge on AUI by transferring the defer-to-boundary technique to groupware systems and investigating it in novel circumstances, as follows.

Firstly, regarding the *type of work*, a group of users would be collaborating, rather than a single user doing individual work, as in every study in Table 7.2. This would allow testing the DTB technique in scenarios where information overload is more likely to occur due to the complexity of group work (see Section 7.1.1 on page 112), namely the higher interruption rate. In addition, if human attention is effectively supported by this technique, the benefits might be experienced not only at the individual level but also at the group level, such as through better user coordination.

Secondly, concerning *task interdependence*, the assumption that the primary and secondary tasks are unrelated—adopted in almost all studies in Table 7.2, with the exception of the first two, in which the user was notified with messages containing predetermined hints—may not hold in group work. In fact, collaboration is usually characterised by intertwined tasks (McDaniel et al., 1996), whose outcomes are dependent on the activities of several users, who aim to contribute to the shared group goal. One possible consequence of this interdependence is that users may feel similar motivation to fulfil the

various tasks at hand and this might cancel the extra time on the secondary task reported in Bailey and Konstan (2006).[6]

Finally, on using AUI to manage *information flows*, the transfer of the defer-to-boundary technique to groupware presents new challenges since its application has so far been confined to feedforward flows that inform users about discrete action possibilities (via notifications or full-screen windows), but computer-mediated cooperative work needs a richer set of information flows (see Section 4.3.1 on page 57).

In particular, the *feedthrough* flow is a good candidate for leveraging the potential benefits of the DTB technique because this information flow is essential for users to maintain an up-to-date awareness of the group, that is, the costs of not paying attention to it are high, and also because it may convey fast-paced information about the actions of the other users, especially with larger groups.

These three differences between collaborative and individual work call for further investigations on the effectiveness of AUI and set the stage for the design and development of novel attentive groupware systems.

7.5 Summary

In this chapter, I argued for the need to evaluate groupware usability regarding information overload because this important problem in our information-rich world is more likely to occur during group work. Thus, I posited that groupware systems should be sensitive to the goals and limitations of human attention, and proposed an approach that highlights the importance of designing for attention scarcity (see Figure 7.1 on page 116).

To this end, I reviewed the state of the art in Attentive User Interfaces but found no evidence of research that uses computers to tackle information overload during collaborative work.

I also noted that several studies addressing individual work in dual-task scenarios suggest interruption management, especially the defer-to-boundary technique, is one way of optimising the user's attention. Consequently, I

6 The conjecture stated in Bailey and Konstan (2006, pp. 697–698) was that users spent more time on the secondary task with the DTB technique because they were not pressed to finish the ongoing primary task had the interruption occurred at a random moment.

proposed transferring this technique to groupware systems and highlighted the novel circumstances and challenges introduced by such a move.

From this situation, I set out to evaluate the usability of a custom-made groupware system that is sensitive to the users' state of attention, which is the subject of the next chapter.

Notes

My first public discussion of the ideas in this chapter took place at the Doctoral Consortium of the twelfth International Workshop on Groupware (CRIWG'06), held in Medina del Campo, Spain.

My first paper concerning the need to address human attention in the design of groupware systems was published in the proceedings of the second edition of the Portuguese *Conferência Nacional em Interacção Pessoa-Máquina (Interacção'06)*, held in Braga (Ferreira and Antunes, 2006b).

Additional research made me aware of the defer-to-boundary technique, which I first mentioned in a presentation at the first Workshop on Adaptation and Personalisation in Social Systems (SociUM'07), held in Corfu, Greece (Ferreira and Antunes, 2007a).

In those two previous papers, I had hypothesised about the development of specialised attentive devices for groupware systems, but subsequent research found that only one of those devices was worth further investigation, which I document in the next chapter.

Finally, the research on the goals and limitations of human attention and a variation of the diagram in Figure 7.1 on page 116 are to appear in the *Group Decision and Negotiation* journal (Ferreira et al., 2010).

Chapter 8

Evaluating an Attentive Groupware System

Having presented the arguments for evaluating groupware usability regarding information overload, and having posited that groups may benefit if groupware systems are designed to support human attention, in this chapter I go from words to action. To this end, I propose a novel attentive groupware device, called *opportunity seeker*, designed to leverage group attention by adapting the defer-to-boundary technique to the ubiquitous fragmented nature of group work.

I explain how the attentive device can be applied to an electronic brainstorming tool that allows users to submit many ideas in parallel, and report on a laboratory experiment, which shows that groups produced more ideas when exposed to the opportunity seeker, thus attesting its contribution to improve groupware usability.

8.1 An Attentive Device for Groupware Systems

To deal with information overload in groupware systems, I considered the limitations of human attention in Section 7.2.2 on page 119, the recommendations for interruption management in Section 7.4, and the groupware information flows in Section 4.3.1, and compiled a set of design guidelines for attentive groupware devices, as follows:

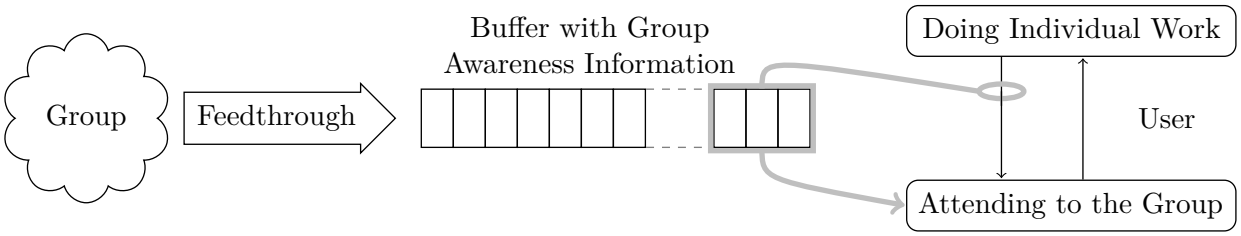


Figure 8.1: Conceptual view of the *opportunity seeker* attentive device. The feedthrough flow that conveys group awareness information is directed to an internal buffer, whose contents are gradually delivered in batches to the user as s/he switches from doing individual work to attending to the group. The opportunity seeker manages one buffer per user.

1. Feedthrough should be intercepted and manipulated as this flow conveys all sorts of information about the actions executed by users and about the state of the group, and, thus, is the most likely source of information overload during group work;
2. Unpredictable bursts of information should be converted into more steady information flows to address our difficulty in detecting multiple items presented in rapid succession, that is, to attenuate the effects of the attentional blink phenomenon;
3. Updates about changes occurring within the group should be bundled together so that users are less frequently interrupted, to make those changes more salient, and to mitigate the effects of change blindness;
4. The delivery of group awareness information should be deferred until task boundaries are reached and these should be assessed independently for each user on the group.

From these guidelines, I devised the *opportunity seeker*, an attentive device for synchronous groupware systems that redirects the feedthrough flow (design guideline 1) to a temporary buffer (guideline 2) and automatically manages the timing and quantity of group awareness information to be delivered to each user based upon his or her attention state (guidelines 3 and 4). A conceptual view of the opportunity seeker is illustrated in Figure 8.1.

There is a trade-off in managing the delivery timing and quantity of group awareness information that reaches users, related to the balance between group focus and distraction (Ellis et al., 1991), which is ingrained in the fragmented nature of group work (Mark et al., 2005):

- Too *few* updates may give the wrong impression about what the group is doing, which reduces the effectiveness of reciprocal task interdependence because the group would more likely engage in conflicting or repetitive actions; and

- Too *many* deliveries may provide up-to-date awareness information but become distracting, which affects the users' capacity to properly attend to the group and increases group coordination overhead.

I address this trade-off by leveraging the typical alternation between users attending to the group and doing individual work to find natural opportunities to interrupt each user. Thus, regarding the delivery timing, the opportunity seeker only sends group awareness information to the user when s/he is likely *not* doing individual work, more specifically, at the transition to paying attention to the group (see Figure 8.1).

This choice of providing feedthrough information between tasks follows from design guideline 4 and, in addition, is aligned with a study about dual-task interference in group support systems (Heninger et al., 2006). This study showed that participants in a text discussion had more difficulties in processing new information because of the need to simultaneously contribute to the discussion. Consequently, the authors proposed the introduction of formal stages so that the users stayed focused on one task at a time.

I note, however, that there is no need to adapt the initial task, as in the previous study, as long as task boundaries can be detected automatically.

Finally, concerning the quantity of information to deliver in a single batch, it should be neither too little (from design guideline 3) nor too much, to avoid overloading the user if his or her work pace differs too greatly from the rhythm of the group.

8.2 Application in Electronic Brainstorming

I implemented the opportunity seeker device on ABTool, a custom-made electronic brainstorming tool with built-in sensors of keystroke-level user activity, to dynamically manage the delivery timing and quantity of ideas sent to each user over brainstorming sessions.

8.2.1 Motivation

Brainstorming is one of the most studied group tasks and this has enabled the identification of many factors that drive creativity gains, such as synergism and encouragement, and losses, including evaluation apprehension (withholding of ideas due to fear of negative opinions from others), production blocking (users forgetting an idea because they had to wait for their turn to speak), and others (Shaw et al., 2002; Nunamaker et al., 1991).

Technology, namely electronic brainstorming, has addressed some of the loss factors by letting users be anonymous to mitigate evaluation apprehension and by letting users submit ideas in parallel, instead of serially as in group turn-taking, to attenuate production blocking (Hymes and Olson, 1992; Connolly et al., 1990).

However, electronic brainstorming tools may also create new conditions that induce creativity losses, in particular *information overload*, which has long been reported in the literature (Grisé and Gallupe, 1999; Nagasundaram and Dennis, 1993; Nunamaker et al., 1991). To see why this may occur, it is necessary to consider two cognitive tasks that follow from the original rules of brainstorming (Osborn, 1963):

1. Produce as many ideas as possible because quantity is wanted; and
2. Read, or at least look at, the other users' ideas because combination and improvement of ideas is sought.

In *electronic* brainstorming users can submit ideas in parallel, which puts more effort in the second cognitive task. As the number of ideas increases, for example because the group is large and productive, users may no longer be able to process the ideas, and may even become distracted by them.

The role of ABTool is precisely to compensate for this type of information overload: as the number of ideas from others increases, the opportunity seeker installed on ABTool stores them in a buffer until it determines the best moment to raise the user's attention with feedthrough.

8.2.2 Preliminary Study

I faced two major challenges in applying the opportunity seeker to ABTool, which I addressed via a preliminary study with users. The challenges were:

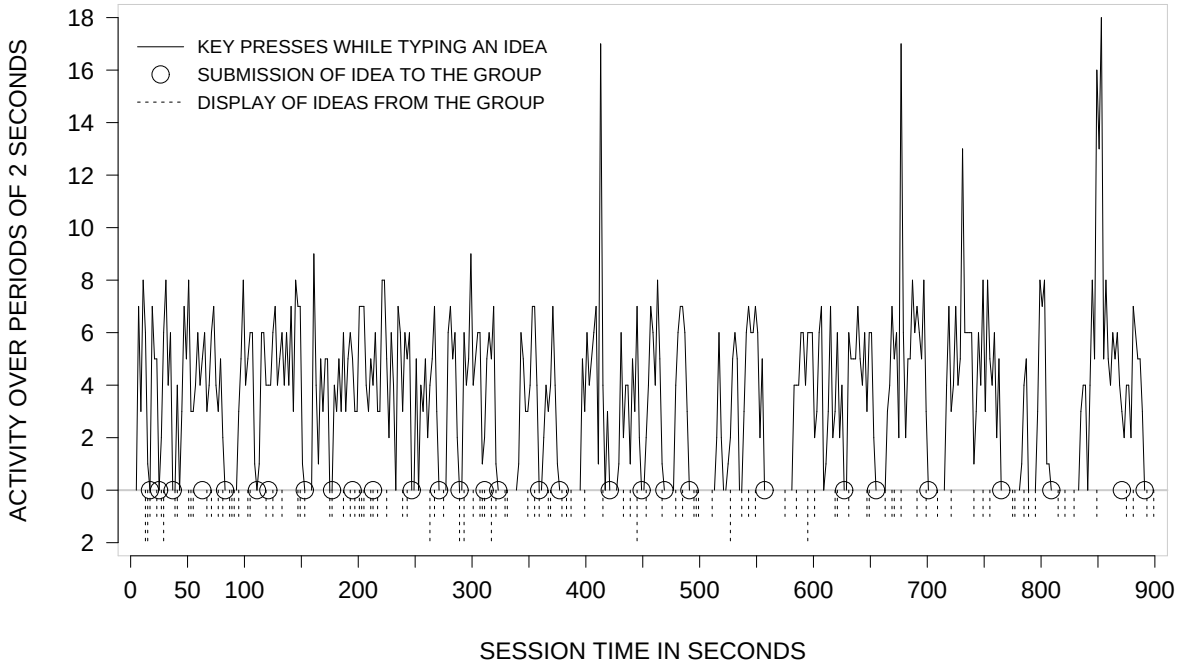


Figure 8.2: User and group activity during a brainstorming session with ABTool, with immediate broadcast of ideas to everyone on the group, without the opportunity seeker. Above the X-axis are aggregated counts of user key presses. The spikes occurred when the user pressed the delete or cursor keys. The circles on the X-axis show when the user submitted the idea s/he was typing to the group. Below the X-axis are the instants in time when the user received ideas written by the other users on the group.

- Characterise how users brainstorm in a scenario were the computer immediately broadcasts new ideas to the group; and

- Find a way to detect task switching during electronic brainstorming, especially between individual work and group attending.

To this end, I gathered groups of five volunteers in a room and asked them to simulate a distributed work setting by using only the tool to communicate, that is, no face-to-face interaction was allowed during the brainstorming sessions. I recorded three types of events: a) user key presses while typing ideas; b) the moments when the user submitted an idea to the group; and c) the instants when group ideas were delivered to each user's computer screen.

Figure 8.2 shows a sample of the data obtained and illustrates the results for an entire fifteen minute session, in which 152 ideas were produced.

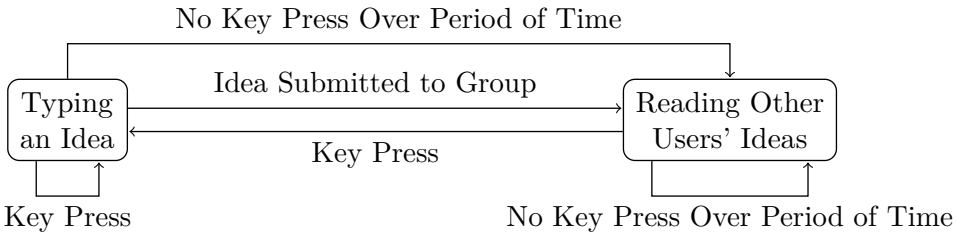


Figure 8.3: Model of user behaviour assumed by the opportunity seeker on ABTool: the user is either typing an idea (doing individual work) or reading other users' ideas (attending to the group). Compare this with the generic model in Figure 8.1.

The evidence collected in this preliminary study was subject to a visual analysis from which three patterns of user activity emerged:

1. Users typically did not stop typing when they received ideas from the other users, thus, I assume they continued focused on the individual task of generating ideas;
2. Users usually stopped typing for a brief period after having put forward an idea, presumably to keep up with the group; and
3. There were numerous periods of time with no typing activity, sometimes lasting more than half a minute (not shown in Figure 8.2).

Regarding the third pattern, I could not tell if the user inactivity was because of lack of imagination or due to free riding, that is users relying on others to do the work (Nunamaker et al., 1991).

8.2.3 Model of User Behaviour

Based upon these three patterns, I hypothesise that a task boundary, that is, an opportunity to deliver ideas from others, occurs when the user submits an idea to the group. In addition, new ideas should be delivered after a period of inactivity (currently, ten seconds) so that the user does not get the impression that the group is not producing ideas too.

Figure 8.3 shows the resulting state transition diagram that models the behaviour of the user as assumed by the opportunity seeker on ABTool.

Another feature of the opportunity seeker on ABTool is that it imposes a limit on the maximum number of ideas from others that can be displayed at once (currently, ten). As I mentioned earlier, this is to avoid overloading the

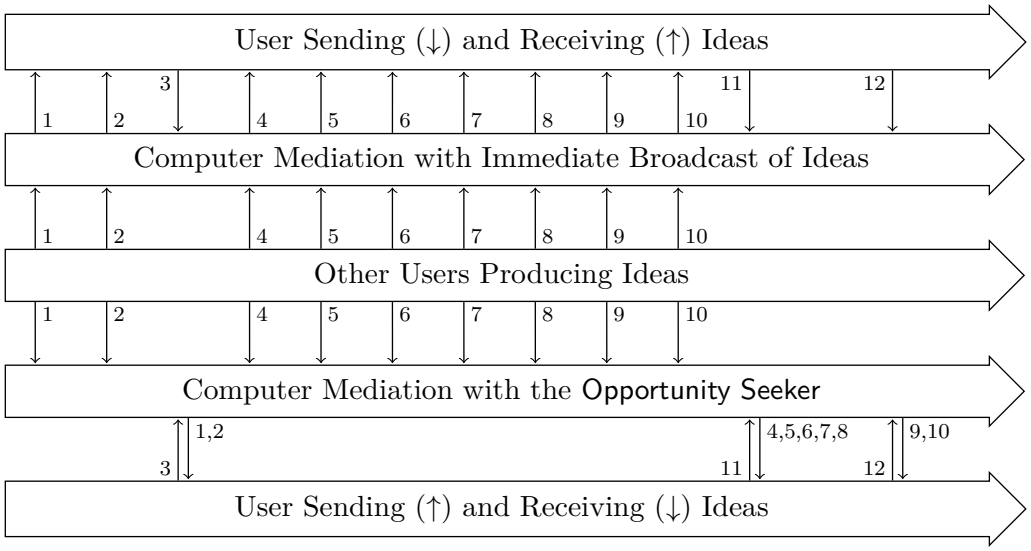


Figure 8.4: Simulation of group and user activity during a brainstorming session with immediate broadcast of ideas (*upper region*) and with the opportunity seeker (*lower region*). In both cases the user produces three ideas (numbered 3, 11, and 12) but his or her exposure to the nine ideas from the other users is different. For illustration purposes, I do not show the propagation of ideas 3, 11, and 12 to the group, and limit the number of ideas in each delivery to five.

user, for example by filling up the entire computer screen with new ideas, when the user is working at a slower pace than the other users. Figure 8.4 shows a simulation that exemplifies the delivery of ideas with the opportunity seeker compared to the immediate broadcast of ideas.

8.2.4 Software Architecture and Design

Technically, ABTool is characterised by a client-server architecture, in which the server manages the delivery of group awareness information via feed-through. The server also collects performance data, which are stored in an XML (eXtensible Markup Language) log. The purpose of the clients, one per user, is to receive input from the users and pass it on to the server, and to display new ideas as they become available from the server.

ABTool is written in C# and is built on top of the Microsoft .NET Framework 2.0. Communication between the clients and the server is done via TCP/IP sockets and all messages, such as ideas, key presses, users joining or retiring the group, sessions starting or ending (see Figure 8.5 for more

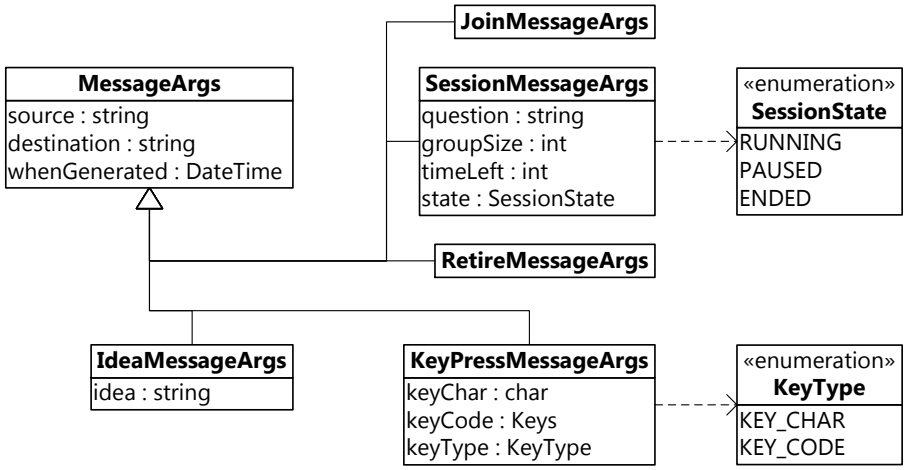


Figure 8.5: Types of messages supported by ABTool, reflecting activity in electronic brainstorming, namely key presses during the typing of an idea, submission of ideas, users joining in and retiring from the group, and definition of the session status, such as the brainstorming question, group size, and time left until it ends.

details) are automatically serialised and deserialised using BinaryFormatter objects attached to NetworkStream instances.

Within the client and server applications, messages are propagated using events, to which consumer objects can subscribe themselves. Given that almost all ABTool classes handle message events, namely the user interfaces, the opportunity seeker, and the classes responsible for receiving and sending messages from/to the network, I defined an IHandlesMessages interface and a default implementation for it, DefaultHandlesMessages. This default implementation class relies on reflection to allow other classes to transparently delegate the determination of the method to run as a function of the type of message associated with the event.

Figure 8.6 shows that the opportunity seeker derives from the AttentiveDevice generalisation, which actually implements immediate delivery of ideas from the users to the group. The OpportunitySeeker class alters this default behaviour by maintaining separate buffers, one per user, containing ideas that have been put forward by the other users. The buffer is stored in the UserNode, which also keeps a Timer object that every verificationPeriod milliseconds verifies the time of the most recent key press by the user, and if it was more than activationTimeSpan milliseconds ago, then it delivers up to ideasAtOnce ideas to the user.

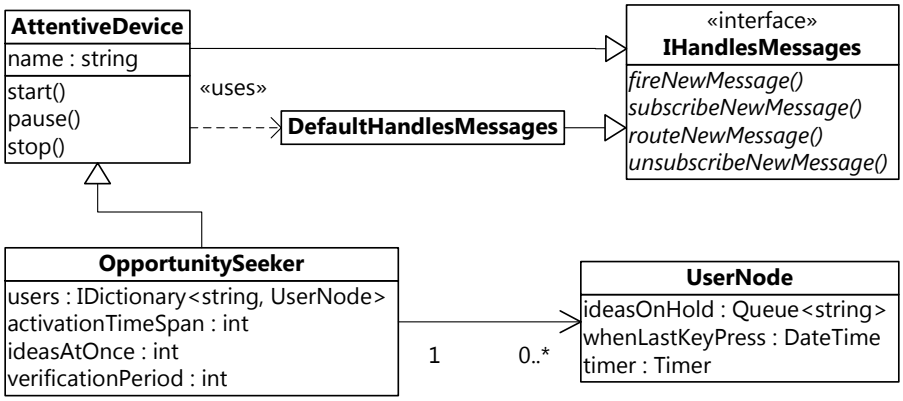


Figure 8.6: Details of the opportunity seeker implementation on ABTool.

The AttentiveDevice and OpportunitySeeker classes implement three methods: start() is run when a session starts or resumes; pause() is executed when, for some reason, the session needs to be paused; and stop() is run at the end of a session. Other methods handle the reception and forwarding of messages, but I omitted those for brevity reasons.

To conclude the presentation of ABTool, I show in Figure 8.7 two screen shots of the client-side interface with the opportunity seeker running, taken in quick succession when the user was finishing typing an idea and submitted it to the group. Note the large shared workspace in the middle of the application window, showing ideas from others, and a smaller private workspace at the bottom, where the user writes one idea at a time.

In conformance with the model of user behaviour in Figure 8.3, as long as the user is typing, he or she is not attending to the group, so ideas from other users are not shown in the shared workspace (see left-hand screen shot in Figure 8.7). However, when s/he submits an idea to the group, the opportunity seeker delivers a batch of ideas from the group that were stored in a buffer, which are shown immediately after the user's own idea (see right-hand screen shot).

8.3 Laboratory Experiment

I now describe a laboratory experiment that I set up to evaluate the usability of ABTool regarding information overload. To do this, I operationalised

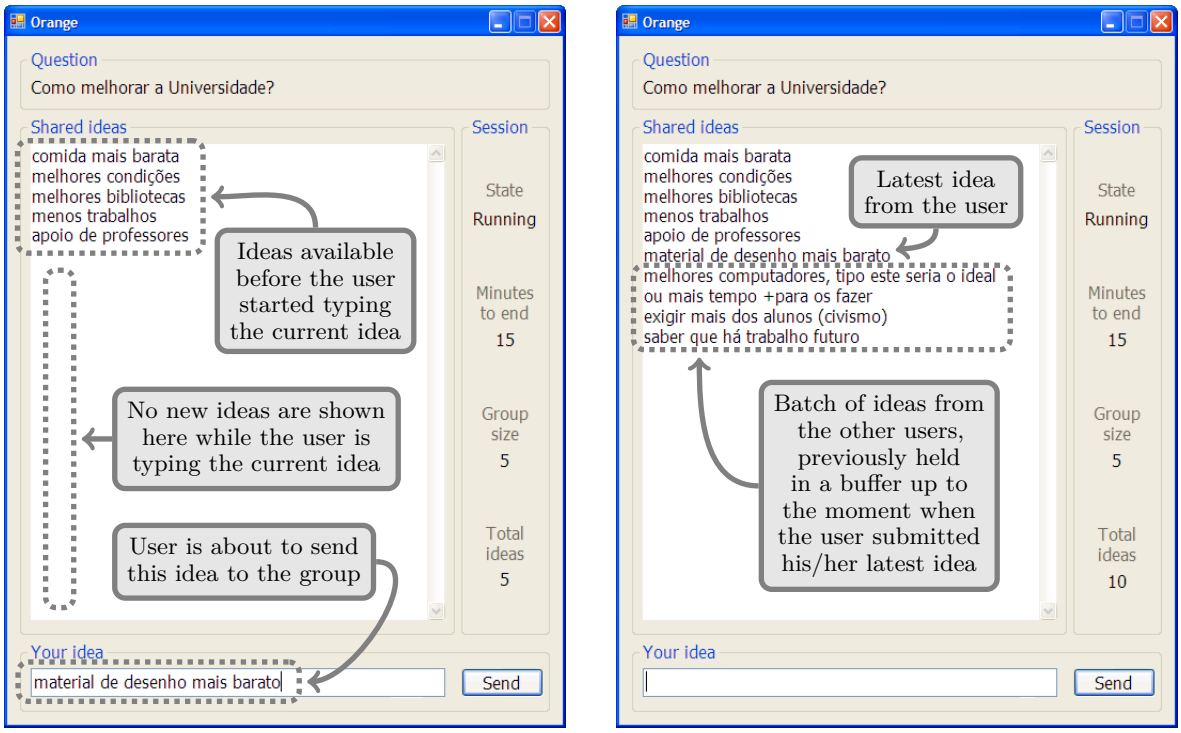


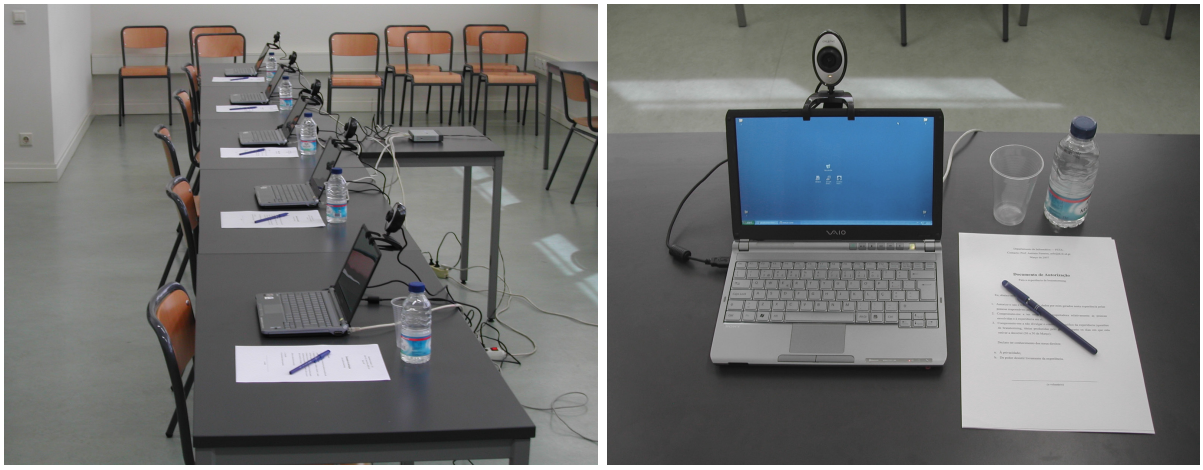
Figure 8.7: Opportunity seeker managing the delivery of ideas on ABTool. *Left:* while typing an idea, the user receives no new ideas from the group. *Right:* when the user submits an idea to the group, new ideas from others are displayed. Note that in ABTool all ideas are anonymous.

groupware usability in terms of the number of ideas produced by the group and hypothesised that if the opportunity seeker mitigates information overload, then the number of generated ideas is greater when groups are under the influence of the opportunity seeker.

8.3.1 Participants

A total of 11 groups of 5 people, for a total of 55 volunteers, 44 men and 11 women, participated in the experiment. The median age was 23 years, ranging from a minimum of 20 years up to a maximum of 29. 51 participants were students (40 undergraduate, 10 MSc, 1 PhD), and the remaining 4 comprised researchers, a software developer, and a translator.

A self-assessment of typing experience with computer keyboards, in a three-point rating scale, revealed that participants were skilled, since 86 % chose the highest score and the remaining 14 % chose the middle score. No one selected the lowest score. This suggests that typing experience was



(a) Laboratory room. (b) Detail of apparatus.

Figure 8.8: Apparatus used for the laboratory experiment. Five tables were placed side-by-side in a row, with one laptop each. The laptops were interconnected by a local wired network and had a Web camera on top of the screen. The paper to the right of each laptop is a consent form.

reasonably *homogeneous* among the participants, which is desirable because otherwise disparities would have likely influenced the speed at which ideas were typed, which would affect the number of ideas produced.

A convenience sampling was used to select participants, who were recruited from social contacts and posters on corridors at the University of Lisbon. No monetary reward was offered and the only information available was that the experiment would concern brainstorming.

8.3.2 Apparatus

The experiment was conducted in a laboratory room having five laptops with identical hardware and software specifications (an Intel Pentium M processor operating at 1.2 GHz, 1 GByte of memory, 60 GBytes of disk, Microsoft Windows XP SP2, .NET Framework 2.0), interconnected by a dedicated 100 Mbit/s Ethernet network.

Figure 8.8(a) shows that the layout of the room where the experiment was conducted comprised a single row with five work desks. I chose this configuration to minimise eye contact between the participants (to reduce distractions) while keeping staging costs manageable. After some training, a team of two was able to make all preparations in about half an hour.

Each laptop had screen-recording software (ZD Soft Screen Recorder 1.4.3), and a Web camera (Creative WebCam Live!) affixed to the top of the screen, which can be seen in Figure 8.8(b). Also, keyboard sensitivity, desktop contents, display resolution, and brightness were controlled. The client application of ABTool was installed on the five laptops.

8.3.3 Task

Participants completed practise and test tasks, consisting of anonymous brainstorming sessions, which were identical in form but different in duration.

The *practise* task allowed participants to get familiar with electronic brainstorming in general and with ABTool in particular. A question was given, namely 'What would you do with an extra finger?,' and then participants were asked to generate as many ideas as possible by typing on the keyboard and by reading other users' ideas on the laptop display. Speech and other forms of communication were disallowed to simulate a distributed work environment and to mitigate extraneous influences.

In the *test* task, the brainstorming session lasted for fifteen minutes, instead of the five minutes for the *practise* task, and the questions were the following four, which I identify with letters: A, How to preserve the environment?; B, How to promote tourism?; C, How to improve the university?; and D, How to stimulate sports practise?[1]

8.3.4 Design

I used a repeated measures design for the experiment (Howell, 2007, pp. 440–441). The independent variable was *device type* and each group of participants was exposed to both a control treatment, with immediate broadcast of ideas to the group, and an experimental treatment, with the opportunity seeker controlling the delivery timing and quantity of ideas to each user (see simulation of these two treatments in Figure 8.4 on page 139). The dependent variable, *group performance*, was calculated from the sum of the number of ideas produced by users on the group per brainstorming session.

1 The practise question and some of the questions in the test task were inspired in the literature; see Shaw et al. (2002, pp. 8–9) for a compilation.

Table 8.1: Session order/brainstorming question per group and treatment. The questions were: A, How to preserve the environment?; B, How to promote tourism?; C, How to improve the university?; and D, How to stimulate sports practise?

	Groups										
Treatment	1	2	3	4	5	6	7	8	9	10	11
Control	1/C	2/D	4/C	3/B	1/B	1/A	2/C	3/B	2/B	3/C	1/A
Experimental	3/B	1/A	2/B	4/C	3/C	2/B	3/A	1/C	1/C	2/A	3/B

The order of exposure to the treatments and the questions used are depicted in Table 8.1, which shows, for example, that group 1 was exposed to the control treatment in the first session, in which ideas for question C were put forward (this corresponds to the top-left cell marked 1/C).[2]

8.3.5 Procedure

A trial started when a group of participants arrived at the laboratory room. Then, a brief introduction to this research was given and participants were informed on their privacy rights and asked to sign a consent form. Next, participants filled out an entrance questionnaire about gender, age, occupation, and typing experience. Written instructions on the rules of brainstorming and on ABTool were then handed in to all participants and read out loud. I provide all these materials in Appendix B on page 171.

After the initial formalities, participants were asked to carry out the practise task for five minutes, after which questions about ABTool were answered. The group then fulfilled the test tasks in succession, each lasting for fifteen minutes, with a brief rest period in between. I chose this session length to stress the importance of generating ideas at a fast pace for a relatively small duration, in which participants would need to remain attentive—interestingly, Dennis et al. (1996) suggest that time constraints increase the rate of idea generation—and also because I wanted to avoid fatiguing the participants due to the nature of the repetitive measures design.

At the end of the trial, answers were given to the questions participants had about this research, comments were annotated, and I gave thanks in acknowledgement of their participation in the experiment.

2 I note that session order is sometimes greater than two and that four questions were used, because I am reporting here a part of a larger experiment with two additional treatments.

Table 8.2: Number of ideas per group and treatment. *M* is the mean, or average, number of ideas per session considering all groups, and *SD* is the corresponding standard deviation.

	Groups													
Treatment	1	2	3	4	5	6	7	8	9	10	11	**Total**	***M***	***SD***
Control	152	83	133	91	264	77	48	53	66	104	70	1141	103.7	62.0
Experimental	192	108	113	117	258	77	68	61	76	116	65	1251	113.7	60.8
Difference	40	25	−20	26	−6	0	20	8	10	12	−5	110	10.0	17.2

8.4 Results

Results are organised in three parts: I start with an analysis of overall group performance, which is central to the hypothesis being tested in the experiment; I then decompose group performance over consecutive periods through the duration of brainstorming sessions; finally, I show results of a *post-hoc* analysis based upon fine-grained data collected at the user level.

8.4.1 Group Performance

Groups produced an average of 9.6 % extra ideas per session, equivalent to 10 additional ideas per session, when under the exposure of the opportunity seeker than when under the control treatment, totalling 1251 versus 1141 ideas for the 11 groups, as shown in Table 8.2. Figure 8.9 shows boxplots for the data in Table 8.2, which further reveals that the difference between treatment medians was 25 ideas per session (83 versus 108).

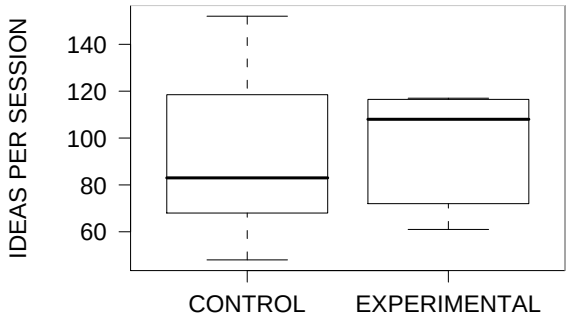


Figure 8.9: Number of ideas produced by groups per session per treatment.[3] Outliers omitted to improve plot legibility. See more details in Table 8.2.

The Shapiro-Wilk normality test indicated that the normality assumption could *not* be accepted for both the control and experimental data distributions ($W = 0.795$, $p = 0.008$; and $W = 0.797$, $p = 0.009$, respectively).[4]

Therefore, I opted for a non-parametric, distribution-free, approach to test the null hypothesis that there is no difference in group performance under the control and experimental treatments. To this end, I applied the Wilcoxon matched-pairs signed-ranks test (Howell, 2007, Sect. 18.7), which revealed a 3.7 % probability of chance explaining the differences in group performance, or $p = 0.037$. In addition, the results $W_+ = 45.5$ and $W_- = 9.5$ from the same test indicate that the biggest differences were due to the experimental scores being greater than the control scores.[5]

I also analysed possible confounding influences from the questions or session order on group performance to see if there was a bias introduced by popular questions or a learning effect due to the nature of the repeated measures design. I applied the Wilcoxon signed-ranks test to both scenarios, which found no significant influences: $p > 0.205$ and $p > 0.343$, respectively.

In these conditions, I can conclude that the results support the hypothesis that groups generated more ideas with the opportunity seeker, which suggests that it contributes to mitigate information overload. I provide more evidence on this in the third part of the analysis, concerning user-level performance.

8.4.2 Group Performance Over Time

Concerning the analysis of group performance through the duration of the brainstorming sessions, I broke down the 900 seconds that each session

3 The rectangles in the boxplots show, from bottom to top, the first, second, and third quartile (or Q_1, Q_2, Q_3) of the data distribution. The second quartile is also the median and is represented by a slightly stronger horizontal line (Howell, 2007, Sect. 2.10). The difference between Q_3 and Q_1 is called inter-quartile range, or IQR. The position of the lower whisker is given by the smallest data value that is greater than or equal to $Q_1 - 1.5 \times IQR$. The upper whisker is given by the largest data value that is less than or equal to $Q_3 + 1.5 \times IQR$.

4 These results were obtained by calling the shapiro.test function in the R software package (`http://www.r-project.org`). Small values of W are evidence of departure from normality. A value of $p < 0.05$ rejects the supposition of normality.

5 I used the wilcox.test R function to obtain these results. This test calculates the differences between the experimental and control scores (as in Table 8.2), and converts them into ranks, where the first rank is for the smallest difference. W_+ is the sum of ranks for positive differences, going for the experimental treatment, and W_- is for the negative differences. In Howell (2007, Sect. 18.7), W_+ and W_- are written T_+ and T_-.

lasted into consecutive periods of 300, 150, and 30 seconds and counted the number of ideas produced during each period. By following this approach, I intended to highlight specific periods when one of the devices would enable better group performance. For example, a brainstorming session may be decomposed into at the beginning, when users usually have plenty of ideas, at the middle, and at the end, when users are typically more passive.

This decomposition is depicted in the top region in Figure 8.10, which shows that in all three periods of 300 seconds groups produced more ideas with the opportunity seeker than with the control device. I obtained similar results at the 150 seconds level of aggregation, as depicted in the middle region in Figure 8.10. Finally, for periods of 30 seconds, groups performed better with the opportunity seeker in 21 out of 30 cases,[6] as shown in the bottom region in Figure 8.10.

From the evidence collected, there seems to be no particular phase when group performance with the opportunity seeker could be considered worse than with the control device.

8.4.3 *Post-hoc* Analysis at the User Level

I also conducted a *post-hoc* analysis based upon the fine-grained user activity data stored in the log files of ABTool (see message types in Figure 8.5 on page 140). The purpose of this analysis was to characterise the users' capability to attend to the others' ideas, comprising multiple aspects of user performance and the deliveries of ideas that they were exposed to, with and without the opportunity seeker.

To this end, I defined *user performance* in terms of the following variables: IDEAS, number of ideas produced per user per session; TIDEA, seconds to write an idea; PAUSE, seconds between a user submitting an idea to the group and restart typing; CIDEA, number of characters per idea; CHARS, total number of characters typed per user per session; and DCHARS, total number of characters deleted per user per session.

6 Assuming the null hypothesis that the opportunity seeker has no greater effect on group performance than mere luck, the Binomial test (Howell, 2007, Sect. 5.9), binomial.test function in R, indicates that there is a 2.1 % probability of chance explaining the 21 out of 30 better results at the 30 seconds period. This can be seen as extra evidence for the experimental treatment, despite the relatively arbitrary selection of periods.

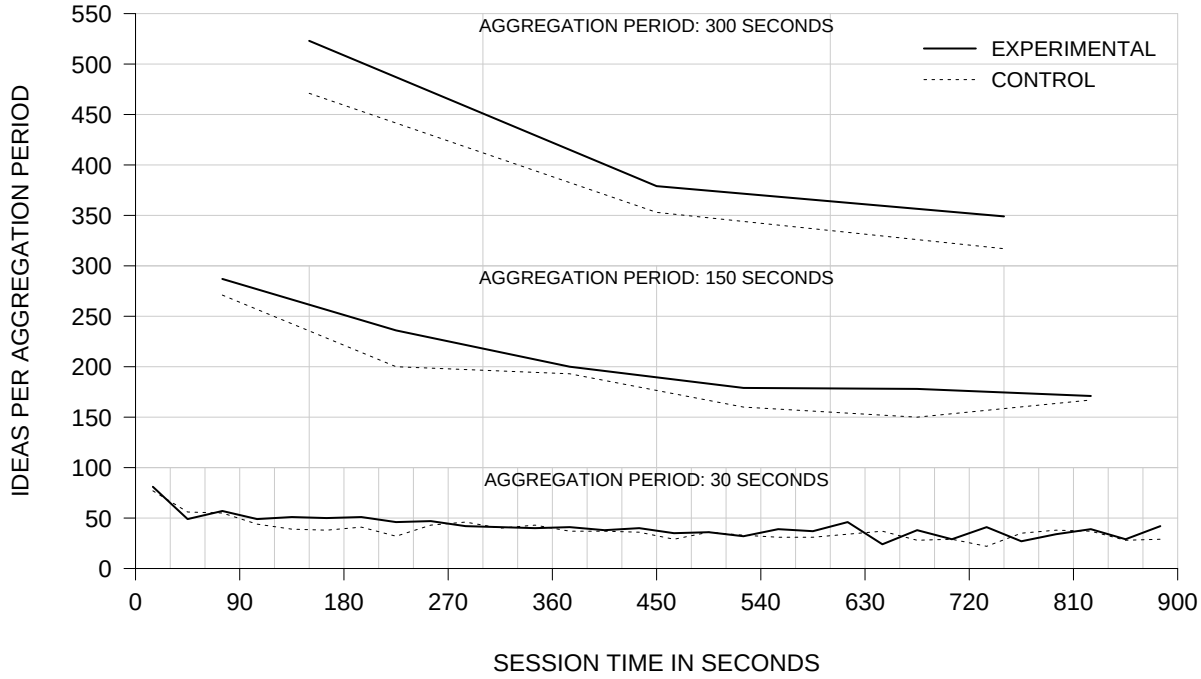


Figure 8.10: Group performance through the duration of the brainstorming sessions under the control and experimental treatments. *Top:* number of ideas per period of 300 seconds. *Middle and bottom:* same, considering periods of 150 and 30 seconds, respectively.

Regarding the characterisation of the *deliveries of ideas* that users were exposed to, I considered variables DLVR, deliveries of ideas per session and TBDL, seconds between consecutive deliveries.

Table 8.3 shows a summary of the results obtained at the user level, including separate descriptive statistics for the two cases in which the users were under the exposure of the control and experimental treatments (creating two large conceptual groups, without and with the opportunity seeker), as well as the output of the Wilcoxon signed-ranks test, which I use here to prioritise the data presentation rather than to do null hypotheses significance testing. Thus, no family-wise corrections were made. Figure 8.11 shows boxplots for the results in Table 8.3, presented in the same order.

Starting with the DLVR variable, the opportunity seeker reduced by an average of 44.1 % the number of deliveries of group ideas that reached a user per session, from a mean value of 82.7 deliveries to 46.2. This difference was due to each delivery having comprised a batch of 1.9 ideas on average ($SD = 1.2$), with up to 5 ideas in 99.8 % of the cases and a maximum batch

Table 8.3: Results of *post-hoc* analysis at the user level, ordered by p-value.

	Control		**Experimental**		**Difference**		**Wilcoxon Test**		
Variable	M	SD	M	SD	M	SD	W_+	W_-	p
DLVR	82.7	48.1	46.2	4.6	−36.5	37.4	0	1540	0.000
TBDL	13.7	5.9	21.2	6.1	7.5	3.2	1540	0	0.000
TIDEA	25.7	17.3	21.5	11.8	−4.2	12.9	422	1118	0.004
IDEAS	20.7	15.0	22.7	13.8	2.0	7.4	929	397	0.006
PAUSE	34.1	34.3	27.7	19.2	−6.4	21.7	469	1071	0.012
CHARS	1044.8	511.2	1110.4	529.8	65.6	321.4	936.5	603.5	0.164
CIDEA	45.6	12.7	43.9	12.9	−1.7	9.5	613	872	0.266
DCHARS	206.7	163.0	199.3	133.3	−7.4	121.9	724	816	0.703

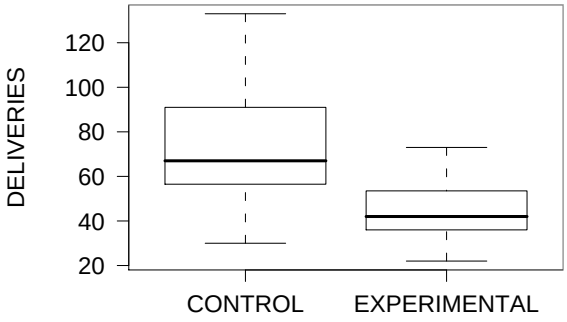
size of 9 ideas (see Figure 8.12), unlike with the control device, which exposed users to immediate broadcasts of ideas, one by one, from the group.

Another consequence of the opportunity seeker, shown in variable TBDL, is that users had 54.7 % more time, on average, to think about and type ideas without receiving new ideas from others, corresponding to uninterrupted periods with a mean duration of 21.2 s versus 13.7 s with the control device.

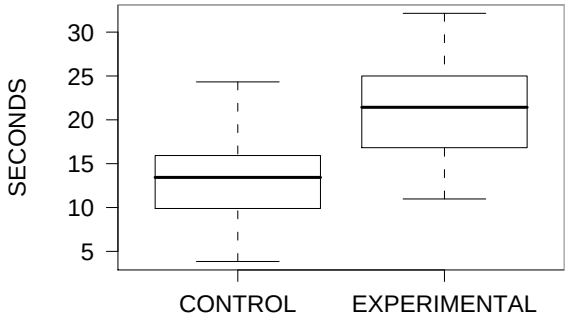
The results for DLVR and TBDL suggest that the opportunity seeker did indeed create the conditions to mitigate information overload by trading up-to-date broadcasts of new ideas for less frequent deliveries of *manageable* batches of ideas. However, this could have penalised the alternation between doing individual work and attending to the group if, for instance, users had slowed down because of the apparent delays in group awareness updates.

In fact, variable TIDEA reveals that users spent an average of −16.3 % of time to write an idea under the experimental treatment, equivalent to a mean cut down of 4.2 s per idea when users typed their ideas without being interrupted with ideas from the group. I also found, through variable PAUSE, that users switched 18.8 % more rapidly, or 6.4 s faster, on average, from submitting an idea to the group to start typing the next idea, presumably reading ideas from others in between.

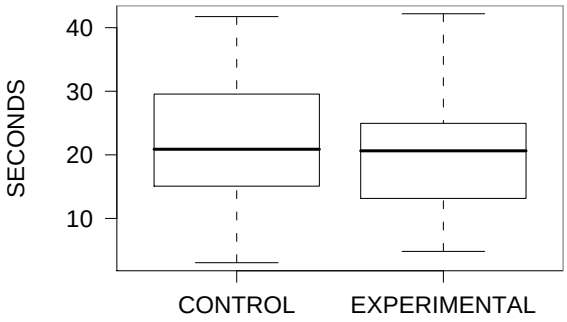
Considering the variable IDEAS, users produced an average of 2.0 extra ideas per session when exposed to the opportunity seeker, corresponding to an improvement of 9.6 %, which was actually expected since this was the percentage of change recorded at the group level (see Section 8.4.1). If there is anything new in this result it is the low p-value of 0.6 %, which further



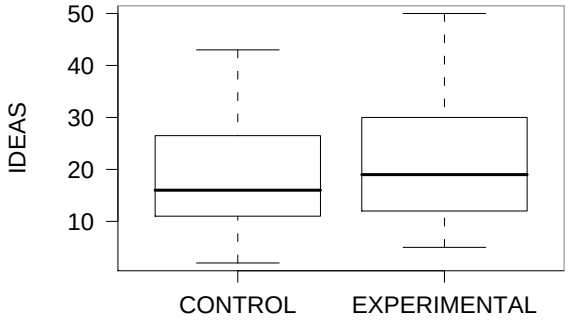
(a) DLVR: deliveries of ideas per session.



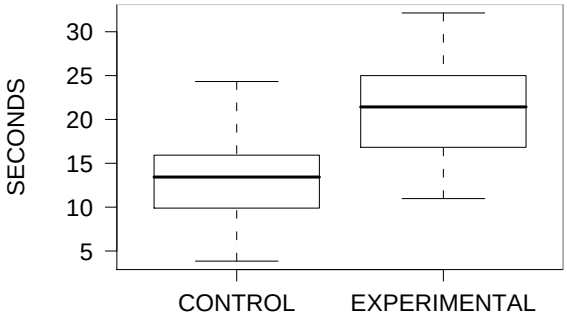
(b) TBDL: seconds between consecutive deliveries.



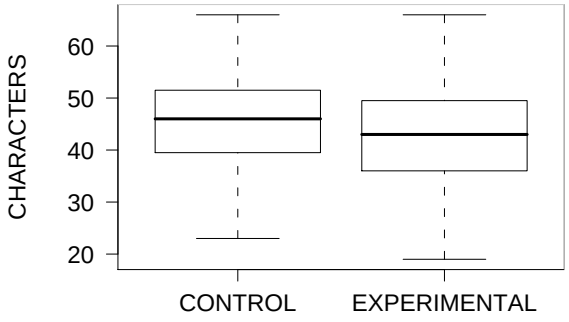
(c) TIDEA: seconds to write an idea.



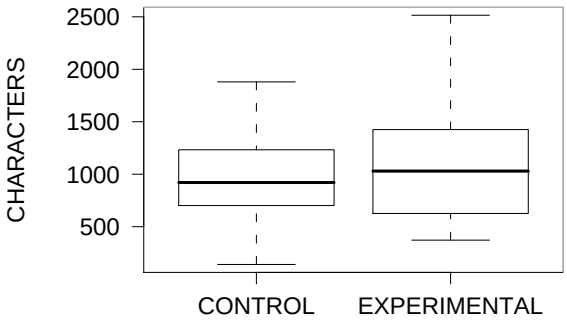
(d) IDEAS: ideas produced per user per session.



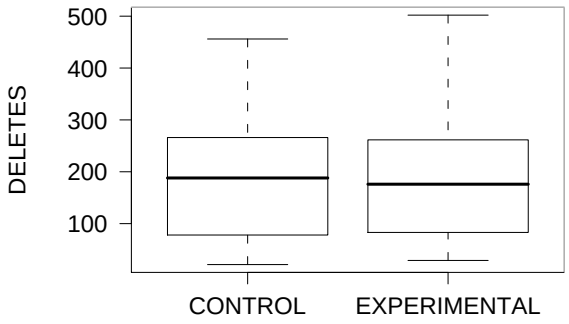
(e) PAUSE: pause between idea submission & typing.



(f) CIDEA: characters per idea.



(g) CHARS: characters typed per user per session.



(h) DCHARS: characters deleted per user per session.

Figure 8.11: Results of *post-hoc* analysis at the user level. See more details in Table 8.3.

Figure 8.12: Distribution of number of ideas per delivery. The maximum number of ideas delivered to a user was nine, and this happened only once.

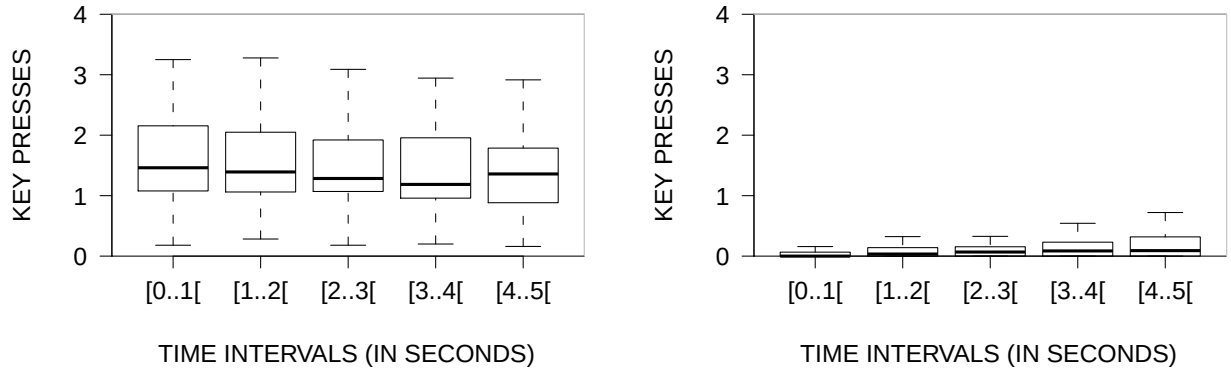
supports the results obtained at the group level, as long as it can be accepted that better user performance implies better group performance.[7]

For the remaining variables in Table 8.3, results revealed smaller differences between the control and experimental treatments, thus likely explained by chance. The number of characters typed per user in a session, CHARS, was 6.3 % higher, on average, with the opportunity seeker, influenced by the higher number of ideas produced (see Table 8.2), but balanced by slightly fewer characters per idea (CIDEA had a mean difference of −3.7 %). Finally, the number of deleted characters, DCHARS, was 3.6 % lower under the experimental treatment, on average.

8.5 Discussion

Having presented the results for group and user performance, which show the opportunity seeker improved the usability of ABTool and which suggest it mitigated information overload in electronic brainstorming, in this section I discuss: a) the validity of the patterns of user activity that regulate the detection of task switching and which were presumed from a preliminary study; b) the adequacy of the operating parameters of the opportunity seeker (batch size and inactivity period), which were *not* derived from theory; c) the potential problem of some ideas not being delivered because of the buffering technique employed; and d) the limitations of the evaluation.

7 This is plausible in brainstorming, where one of its rules is that users should produce as many ideas as possible to further encourage group creativity (see page 136).



(a) Pattern 1.

(b) Pattern 2.

Figure 8.13: Typing activity with immediate broadcast of ideas. The first and second patterns of user activity, which were originally identified through visual analysis (see page 138), are supported by the data gathered in the experiment. *Left:* users did not stop typing during the five seconds following the reception of an idea from the group (pattern 1). *Right:* typing activity was almost nonexistent in the five seconds coming after the submission of an idea to the group (pattern 2).

8.5.1 Validity of Patterns of User Activity

Earlier, I identified three patterns of user activity in electronic brainstorming with immediate broadcast of ideas from the visual analysis of plots such as the one shown in Figure 8.2 on page 137. These patterns are important because they supply the basis for the model of user behaviour depicted in Figure 8.3, which determines the moments when group awareness information should be delivered to the user.

I now provide additional evidence for patterns 1 and 2—that users typically do not stop typing when they receive new ideas from the group and that users usually pause after putting forward an idea, respectively (see also page 138)—based upon fine-grained data stored in the log files of ABTool and gathered throughout the laboratory experiment (see Figure 8.13).

On the one hand, in the five seconds following the reception of new ideas from the group, users continued typing their idea at a mean rate between 1.4 and 1.6 key presses per second (*SD* between 0.7 and 0.8). On the other hand, following the submission of an idea to the group, users almost stopped typing for at least five seconds, with a mean key rate of between 0.1 and 0.2 key presses per second (*SD* between 0.2 and 0.3). This should provide enough evidence to validate the two patterns mentioned above.

8.5.2 Batch Size and Inactivity Period

These two operating parameters are at the core of the opportunity seeker and, as alluded in Section 8.2.3 on page 138, their values on ABTool were *not* derived from theory but, rather, are educated guesses from common-sense.

Considering with the maximum batch size of 10 ideas per feedthrough delivery, I do not have strong evidence regarding this decision. However, I note that the average number of ideas per delivery during the experiment was 1.9 and that 99.8 % of all deliveries had 5 ideas or less (see text near Figure 8.12). These results seem to indicate that the specified batch size actually had no impact on the experiment, and indeed may be of secondary importance for small groups.

However, care should be taken with large groups. For instance, if a group has 20 or more users, and assuming an average rate of 1.5 ideas per minute per user and a mean time to write an idea of 21.5 seconds,[8] then the Poisson probability (Taha, 2003, Ch. 17) of more than 10 ideas arriving during the time to generate an idea would be greater than 50 %. In other words, the number of ideas arriving would likely be greater than the batch size, causing increasing delays in the delivery of ideas, and ultimately leading to several of them not being presented to the users. This could hamper group creativity and productivity, although the benefits of delivering those extra ideas in larger batches (at a rate of 30 per minute) would certainly be outweighed by the cost of information overload.

Regarding the inactivity period of 10 seconds that triggers the next delivery of a batch of ideas to a user, again, I do not have strong evidence to support this decision. However, the data in Table 8.3 reveals that the average pause duration when participants were exposed to the control treatment was 34.1 seconds. This might indicate that the inactivity period should be *longer* because it may well be that users need more than the established 10 seconds to think about an idea without being interrupted.

Interestingly, the same data shows that with the opportunity seeker the average pause duration was reduced to 27.7 seconds, which suggests that it leveraged the users' alternations between doing individual work and paying attention to the group, as intended.

8 These data were taken from Table 8.2 and Table 8.3, for the experimental condition.

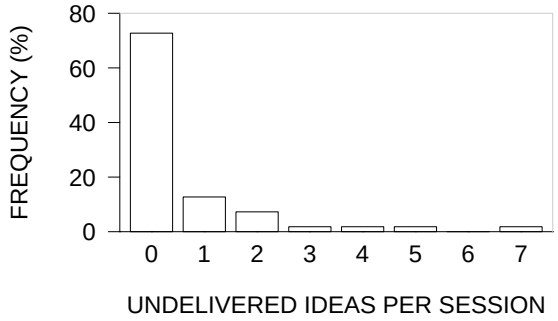


Figure 8.14: Distribution of number of undelivered ideas at the end of brainstorming sessions. A value of 0 means the opportunity seeker delivered all ideas.

8.5.3 Undelivered Ideas

One of the concerns of buffering ideas during brainstorming sessions, instead of immediately broadcasting them, is that ideas submitted near the end of the session may not be delivered to some of the users. This may happen if a user is less productive than the others, either because s/he types very slowly or does not type at all due to lack of inspiration. In these circumstances the opportunity seeker delays the delivery of new ideas from others, limited to a predefined quantity, until the user finally submits the idea to the group or until a timeout occurs (see also Section 8.2.3 on page 138).

To measure undelivered ideas, I subtracted the number of ideas delivered to users by the opportunity seeker from the number of ideas produced by the group. Figure 8.14 shows that in 72.7 % of the cases (40 sessions out of 55) all ideas were delivered to users and that in 20.0 % of the times one or two ideas were not delivered. The remaining 7.3 % were for cases with between three and seven undelivered ideas, each occurring in only one session.

In other words, the users' natural work rhythm was rapid enough so that less than one idea ($M = 0.6$, $SD = 1.4$) was not delivered at the end of a session with the opportunity seeker, which seems reasonable.

8.5.4 Limitations

Recalling the problem identified at the beginning of this chapter, the main objective of this evaluation was to determine if information overload would be mitigated by a groupware device designed to augment human attention.

Ideally, such an evaluation should be based upon direct measures of information overload and human attention, and upon measurements taken in identical tasks with and without the attentive device.

I did not follow this path, however, because, to the best of my knowledge, those *direct* measures do not exist. Instead, researchers have been using indirect or surrogate measures, namely: performance measures in primary and secondary tasks, subjective measures, and physiological measures (Wickens and McCarley, 2008, pp. 120–123; Wickens and Hollands, 2000, pp. 459–470). All these types of surrogate measures pose challenges to the researcher:

- It is common sense that primary task performance may improve for reasons not related to information overload. Also, the measure used is dependent of the context, which makes it harder to do comparisons;
- Secondary task performance is grounded on the fulfilment of standard tasks[9] which may interfere with the execution of the primary task;
- Subjective measures are based upon self-assessments of the perceived difficulty of doing a task, which may be influenced by other factors, such as dislike of or unfamiliarity with the task; and
- Physiological measures require specialised equipment (see some examples on page 124), which may be hard to obtain, may impose physical constraints to the users, and may produce quantities of data that are hard to manage, especially with multiple users on a group.

The path I have chosen for the evaluation is, thus, based on primary task performance measures, meaning that, in the end, I can not assess to which degree users were subject to information overload, if any.

Nonetheless, considering the electronic brainstorming sessions with AB-Tool, the number of ideas produced by groups when exposed to the opportunity seeker was higher, which suggests that more cognitive resources were available to generate ideas and to attend to other users' ideas, and even more so because the time to type ideas without being interrupted also increased by action of the opportunity seeker.

9 Examples of standard secondary tasks include: generating random numbers, reacting to probes, production of constant time intervals, and retrospectively estimating elapsed time (Wickens and Hollands, 2000, pp. 463–464).

Another limitation of the evaluation is that the three patterns of user activity in Section 8.2.3 on page 138 could not be observed in the video feeds of the computer screen and the users' faces (see apparatus in Section 8.3.2 on page 143). I was particularly interested in finding out:

- If users are able to attend to other users' ideas and simultaneously write their own ideas, which would challenge the distinction between the two states in the model of human behaviour assumed by the opportunity seeker on ABTool (see Figure 8.3 on page 138);

- If the pause in typing activity after the submission of an idea coincides with the user looking at others' ideas, which would confirm the corresponding transition between the two states in the same model of human behaviour; and

- If periods of inactivity correspond to lack of imagination, distraction, or engaged reading, which would better inform the appropriate moment to interrupt the users with additional ideas from the group.

However, the videos showed users who appeared to be focused on the task and computer screen most of the time. Very occasionally, there was an outward reaction to reading an idea, such as a frown or a smile, and it was also infrequent to observe users acting distracted, for example, staring somewhere else than the computer screen. Thus, it was not possible to accurately distinguish when a user was reading ideas, pausing, or distracted, so these data had to be discarded.

Concerning the operating parameters of the opportunity seeker, they remained constant for the entire experiment: no more than ten ideas were delivered at once and the inactivity period after which ideas would be sent to the user was ten seconds. I could have tried other values but that would have increased the complexity of the experimental design beyond available logistics possibilities.

Finally, the evaluation excluded an analysis of the quality of the ideas, an ever present topic in brainstorming research. Then again, quantity is one the goals of brainstorming (Osborn, 1963) and it is mainly because of quantity that information overload is likely to occur.

8.6 Summary

In this chapter, I evaluated the usability of an attentive groupware system, which features a device that intercepts feedthrough and applies buffering of group awareness information to mitigate information overload (see Figure 8.1 on page 134). The attentive device, called opportunity seeker, considers the natural rhythms of group work to time the delivery of awareness information with the situations in which users are most likely to benefit from them.

I showed how this device can be implemented on an electronic brainstorming tool and how task boundaries can be detected via keystroke-level activity. I provided evidence that the opportunity seeker increased the work done by groups, and that the improvement amounts to 9.6 % in the number of ideas produced in electronic brainstorming tasks.

In addition, results from a post-hoc analysis show that the opportunity seeker reduced the number of deliveries of ideas by 44.1 % by combining ideas in small batches and that this translated into 54.7 % more time for users to think about and type ideas without receiving new ideas from others. In these conditions, users were 18.8 % faster in alternating between generating an idea, which they did in 16.3 % less time, and reading other users' ideas.

Given these results, I can conclude that focusing the evaluation on human information processing limitations can improve groupware usability.

Notes

The work in this chapter has reached three distinct communities of researchers, as evidenced by the published papers. One is the group decision and negotiation community, which has particularly welcomed the laboratory experiment with ABTool. I presented a paper at the ninth Meeting on Group Decision and Negotiation (GDN'08), held in Coimbra, in which I focused on the brainstorming task and did not mention any of the ABTool internal details (Ferreira and Antunes, 2008a). An improved version of that paper, which reflects much of the recent work in this chapter, is to appear in the *Group Decision and Negotiation* journal (Ferreira et al., 2010).

Another paper about ABTool, which included a more elaborate discussion about the implications of using a buffering technique during group work,

was presented by my adviser at the fourteenth International Workshop on Groupware (CRIWG'08), held in Omaha, USA (Ferreira et al., 2008). It was with this community that, two years before, in the Doctoral Colloquium, I presented the first results of an experiment with attentive groupware devices.

Finally, the inspiration for the model of user behaviour came during the preparation of the paper that I presented at fifteenth International Workshop on Design, Specification, and Verification of Interactive Systems (DSV-IS'08), held in Kingston, Canada (Ferreira and Antunes, 2008b).

Chapter 9

Conclusion

This dissertation has explored the importance of the cognitive level of human action in the evaluation and improvement of groupware usability. The research was motivated by the problem that current methods target the rational and social levels of human action and yet an increasing number of users is relying on computers to fulfil quick and repetitive collaborative tasks that are dominated by perceptual, cognitive, and motor skill. Therefore, there is increasing value in optimising groupware usability at the cognitive level because even small improvements can have large net effects.

In addition, our information-rich world imposes considerable demand on our limited information processing capabilities, especially on users doing group work as they have to attend to multiple information flows and thus are more likely exposed to information overload, which further justifies doing usability evaluations at the cognitive level of human action.

My research perspective has been that of the groupware designer who seeks to improve the usability of collaborative tasks done through the computer by conducting cognitive-level evaluations of human performance in routine and potentially fast-paced scenarios of collaboration.

This chapter concludes the dissertation and has three parts. In the first, I revisit the research goals stated in Chapter 1 and present my contributions to the groupware body of knowledge. In the second part, I summarise the lessons I learned during the investigations, hoping that other practitioners find them valuable. In the third, I describe opportunities for future work along the vision that guided this research.

9.1 Main Contributions

The *research question* underlying this dissertation was how to evaluate and improve groupware usability at the cognitive-level of human action. To address it, I stated three objectives which I now consider alongside the contributions of my work.

9.1.1 Model of the Groupware Interface

The *first objective* of this work was to show that the cognitive level of human action can be useful to organise the design space of groupware interfaces.

I determined that I would create a groupware model grounded on cognitive-level information processing, and that this objective would be met if the model can be applied in a range of groupware systems and if it can be integrated in usability evaluations.

The *first contribution* of this research is the groupware interface model described in Chapter 4. The model leverages existing knowledge about human behaviour with single-user systems by expanding its application to multi-user systems. To do this I have shown that the fundamental differences between users interacting with the computer to do individual work and interacting with other users through the computer to collaborate can be supported by specialised groupware information flows and input/output devices.

The model is instrumental in the first step of the method for evaluating shared workspace usability in Chapter 5, and I have applied it to define the interface of three distinct groupware systems (see Section 5.3 on page 74). I have also referred to the model in the description of the opportunity seeker attentive device in Section 8.1 on page 133. This provides evidence for the accomplishment of the first objective.

9.1.2 Cognitive-Level Groupware Evaluation Methods

The *second objective* was to show that cognitive-level evaluations can predict the usability of collaborative tasks done through groupware systems.

I determined that I would construct two groupware usability evaluation methods, differing in the way usability is measured and in the type of collaboration supported, both grounded on existing engineering models of

human performance, and that this objective would be met if the methods can contribute to formative evaluation of groupware usability in a variety of scenarios of collaboration.

The *second contribution* of this dissertation is a pair of methods for evaluating the usability of groupware systems at the cognitive level of human action. The first method, presented in Chapter 5, applies to critical scenarios of collaboration occurring routinely in shared workspaces and defines usability in terms of time to execute collaborative tasks, as predicted by engineering models of human performance.

I have used this method to evaluate and compare the usability of competing designs in three distinct cases of shared workspace activity.

The second method, in Chapter 6, aims at capturing the intertwined nature of mixed-focus collaboration, encompassing shared and private workspaces, as well as the often conflicting goals of users working as individuals or as members of a group. To do this, I have combined predicted task execution times with task contributions to group progression towards a common goal, in terms of individual productivity, opportunities created for others, and restrictions to the work of the other users.

I have applied this method in a collaborative game, and have shown that the method challenges the groupware designer to think about the trade-offs between an interface that allows users as individuals to be more productive and another that permits greater overall group performance.

The two evaluation methods do not require user testing or functioning groupware prototypes to produce usability results, which attests their formative nature, and facilitates their integration into the prevailing interactive system design process. Thus, the second objective is met.

9.1.3 Evaluation of an Attentive Groupware System

The *third objective* was to show that focusing the evaluation on human information processing limitations can improve groupware usability.

I determined that I would design and implement a device that automatically adjusts collaborative information flows according to each user's state of attention to mitigate information overload, and that I would carry out a laboratory experiment to compare usability with and without the device in a

fast-paced collaborative task, and I also decided that the objective would be met if the experiment is valid and produces statistically significant results.

The *third contribution* of this research is the evaluation of an attentive groupware system, in Chapter 8, which implements a novel attentive device, called the opportunity seeker, that automatically adjusts the delivery of group awareness information according to users' natural task switching between doing individual work and attending to the group. I have shown how this device can be implemented on an electronic brainstorming tool, called ABTool, and how task boundaries can be detected via the keyboard.

I have conducted a laboratory experiment with ABTool in which I told users to submit ideas in parallel as fast as possible, and collected evidence that when groups were exposed to the opportunity seeker the number of ideas produced increased by 9.6 % when compared to the condition in which ideas were immediately broadcasted to all users.

I also have carried out a *post-hoc* analysis showing that the opportunity seeker reduced the number of deliveries of ideas by 44.1 % since it combined ideas in small batches, and that this translated into 54.7 % more time for users to type ideas without being notified of other users' ideas. In these conditions, users were 18.8 % faster in switching between writing an idea, which they did in 16.3 % less time, and reading ideas from the other users.

These results indicate that focusing the evaluation on human information processing limitations has lead to improvements in groupware usability, which meets the third objective.

* * *

With this set of contributions, I have shown that the cognitive level of human action plays an important role in groupware usability evaluation, complementing the rational and social levels that have traditionally been the focus of other evaluation methods.

9.2 Lessons for Practitioners

One thing I learned in this research is that groupware systems have the power to shape new ways for people to collaborate. The trends I mentioned in Chapter 1 indicate that the Web is fast becoming a prime platform for

collaboration, complementing or even replacing the traditional physical workplace. Of course, millions of people already use the Internet to communicate with others, share content, and play multi-player games.

In these circumstances, if a denominator has to be found upon which to do usability evaluations of collaborative tasks done through the groupware interface it is the cognitive level of human action, because it is largely unaffected by the social, cultural, and organisational heterogeneity that exists in on-line communities. So, the *first lesson* is that it is increasingly important to invest in evaluations of quick and routine collaborative tasks executed by a growing number of users because even small improvements in usability can have large net effects. I show how this can be done with the methods in Chapter 5 and Chapter 6.

Another thing I learned is that groupware systems should be evaluated regarding information overload, a contemporary problem in our information-rich world, which worsens as the needs for collaboration rise. This is because, as I mentioned in Chapter 7, users doing group work have to divide their attention between doing individual work and keeping up with the group, which by itself is a potential generator of large quantities of information.

One way of tackling this problem is by buffering group awareness information and releasing it in convenient batches according to each user's state of attention. This approach alters the way groupware information flows are managed and requires sensors of human attention, which can be as simple as keyboard activity, but it successfully contributed to improve the usability of an electronic brainstorming tool. Thus, the *second lesson* is that there is a need for attentive groupware systems and I hope that many more appear in the future, perhaps created along the guidelines I present in Section 8.1.

9.3 Future Work

Future work will carry on the research on attentive groupware systems under project Attentive CSCW, PTDC/EIA/67589/2006, funded by the Portuguese Foundation for Science and Technology. In fact, the opportunity seeker, described in Section 8.1 on page 133, is the most successful of a total of four devices that I have already tested with groups of volunteers, and I

will continue to do experiments with new devices and new groupware tools.

By following this path I will address many questions raised by this research: will the opportunity seeker improve the usability of other types of collaborative tasks done through computers? For instance, will it be useful in convergence tasks, such as in the group negotiation scenario described in Section 5.3.3 on page 85, where users can post arguments in parallel, thus potentially causing information overload? What other devices may be created to augment human attention in group work settings? The road is open for many more experiments and applications.

Appendix A

Equation for Average Number of Viewport Moves

In this appendix, I present the equation for $\overline{m}$, which plays a role in the analytical evaluation of shared workspace performance of the critical scenario 'locating updated objects,' in Section 5.3.1, page 75. The purpose of $\overline{m}$ is to compute the average number of viewport moves to reach an arbitrary object that is not currently visible on the computer display[1] but is located elsewhere on the shared workspace.

To specify $\overline{m}$, I considered the relevant assumptions about the shared workspace described on page 78, namely that its size is multiple of the size of the computer display (assumption 2), the uniform distribution of updated objects (assumptions 3 and 4), and the navigation around the workspace (assumptions 7 and 8). Concerning the latter point, the cursor keys can be used to move the viewport a screen-full to the desired direction and it is possible to go directly from one extreme region to the opposite side (for example, from the leftmost to the rightmost region) of the shared workspace.

In the original assumptions, the shared workspace is a larger version of the computer display, that is, it has the same width × height proportion, only enlarged by a factor of n. Here, I consider a more general purpose equation,

1 The viewport takes up the entire space of the computer display.

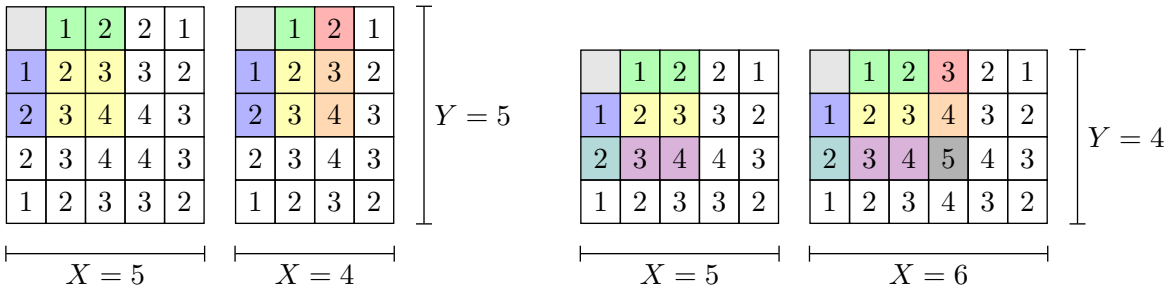


Figure A.1: Shared workspaces with even and odd X and Y sizes. The numbers in the cells indicate the minimum number of viewport moves to reach those regions of the shared workspace starting from the top left corner (with grey background), where the viewport is currently located.

one in which the shared workspace may be a rectangle of any proportion, so I replaced n with integers X and Y. Figure A.1 shows examples of several shared workspaces; for instance, the rightmost one is six times wider than the computer display ($X = 6$) and four times higher ($Y = 4$).

Each region of the shared workspace is annotated with integers that indicate the minimum number of viewport moves to reach objects in those regions, starting from the top-left corner, where the viewport is located. The colours indicate areas of interest with common properties, which are useful for the computation of $\overline{m}$. Actually, each of the four shared workspaces in Figure A.1 represent a special case, which I analyse independently.

Starting with the case in which both X and Y are odd integers, the three areas of interest are highlighted in Equation A.1, which adds that there are four areas equal to the yellow area (see leftmost example in Figure A.1), and that both the green and violet areas have one replica, located in the topmost row and leftmost column, respectively. The terms inside the big round parentheses compute the sum of all possible viewport moves, which is divided by the total number of regions on the shared workspace (except the one region shown by the viewport) to give the average, $\overline{m}$.

$$\begin{aligned}\overline{m}(X,Y) &= \left(4\left[\sum_{x=1}^{\frac{X-1}{2}}\sum_{y=1}^{\frac{Y-1}{2}}(x+y)\right] + 2\left[\sum_{x=1}^{\frac{X-1}{2}} x\right]\right.\\ &\qquad \left. + 2\left[\sum_{y=1}^{\frac{Y-1}{2}} y\right]\right) \div (XY-1)\\ &= \frac{X+Y}{4} \qquad \text{if } X \text{ is odd, } Y \text{ is odd.} \end{aligned} \tag{A.1}$$

Equation A.2 and Equation A.3 can be used for cases where one of X or Y is odd and the other is even, which introduces two more areas of interest, making a total of five different terms inside the big round parentheses (see two examples nearest to the centre of Figure A.1).

$$\begin{aligned}\overline{m}(X,Y) = &\left(4\left[\sum_{x=1}^{\frac{X}{2}-1}\sum_{y=1}^{\frac{Y-1}{2}}(x+y)\right] + 2\left[\sum_{y=1}^{\frac{Y-1}{2}}\left(\frac{X}{2}+1+(y-1)\right)\right]\right.\\ &\left. + 2\left[\sum_{x=1}^{\frac{X}{2}-1} x\right] + \frac{X}{2} + 2\left[\sum_{y=1}^{\frac{Y-1}{2}} y\right]\right) \div (XY-1)\\ = &\frac{X(Y^2+XY-1)}{4(XY-1)} \qquad \text{if } X \text{ is even, } Y \text{ is odd.} \end{aligned} \tag{A.2}$$

$$\begin{aligned}\overline{m}(X,Y) = &\left(4\left[\sum_{x=1}^{\frac{X-1}{2}}\sum_{y=1}^{\frac{Y}{2}-1}(x+y)\right] + 2\left[\sum_{x=1}^{\frac{X-1}{2}}\left(\frac{Y}{2}+1+(x-1)\right)\right]\right.\\ &\left. + 2\left[\sum_{x=1}^{\frac{X-1}{2}} x\right] + 2\left[\sum_{y=1}^{\frac{Y}{2}-1} y\right] + \frac{Y}{2}\right) \div (XY-1)\\ = &\frac{Y(X^2+XY-1)}{4(XY-1)} \qquad \text{if } X \text{ is odd, } Y \text{ is even.} \end{aligned} \tag{A.3}$$

Equation A.4 is for the case where both X and Y are even, which needs to take into consideration a total of eight different areas of interest (see rightmost example in Figure A.1).

$$\begin{aligned}\overline{m}(X,Y) = &\left(4\left[\sum_{x=1}^{\frac{X}{2}-1}\sum_{y=1}^{\frac{Y}{2}-1}(x+y)\right] + 2\left[\sum_{x=1}^{\frac{X}{2}-1}\left(\frac{Y}{2}+1+(x-1)\right)\right]\right.\\ &+ 2\left[\sum_{y=1}^{\frac{Y}{2}-1}\left(\frac{X}{2}+1+(y-1)\right)\right] + \left[\frac{X}{2}+\frac{Y}{2}\right] + 2\left[\sum_{x=1}^{\frac{X}{2}-1} x\right]\\ &\left. + \frac{X}{2} + 2\left[\sum_{y=1}^{\frac{Y}{2}-1} y\right] + \frac{Y}{2}\right) \div (XY-1)\\ = &\frac{XY(X+Y)}{4(XY-1)} \qquad \text{if } X \text{ is even, } Y \text{ is even.} \end{aligned} \tag{A.4}$$

For convenience, the complete set of equations to compute the average number of viewport moves to locate an object on the shared workspace, considering all combinations of odd/even X and Y, is gathered in Equation A.5.

$$\overline{m}(X,Y) = \begin{cases} 0 & \text{if } X = 1,\ Y = 1, \\ \dfrac{X+Y}{4} & \text{if } X \text{ is odd, } Y \text{ is odd,} \\ \dfrac{X(Y^2+XY-1)}{4(XY-1)} & \text{if } X \text{ is even, } Y \text{ is odd,} \\ \dfrac{Y(X^2+XY-1)}{4(XY-1)} & \text{if } X \text{ is odd, } Y \text{ is even,} \\ \dfrac{XY(X+Y)}{4(XY-1)} & \text{if } X \text{ is even, } Y \text{ is even.} \end{cases} \tag{A.5}$$

As a final note on this topic, the data in Figure 5.3 on page 80 were calculated by applying Equation A.5 to cases where $X = Y = n$, with n ranging from two to nine.

Appendix B

Materials Used in the Experiment

This chapter contains samples of the materials used in the electronic brainstorming experiment, described in Chapter 8.3 on page 141. A total of three documents were handed in to all participants in each trial of the experiment: the first is the consent form, which had to be accepted and signed before a participant could be admitted; next is the entrance questionnaire, to characterise the participants while preserving their privacy; and finally, the instructions about brainstorming in general and about the electronic tool that would be used during the trail. For more details about the experimental procedure in the laboratory room, see Section 8.3.5 on page 145.

Please note that the materials are written in Portuguese because the experiment took place in Portugal. I have chosen to keep the original versions in the illustrations and do the translations to English in the text.

B.1 Consent Form

Figure B.1 exhibits the consent form that all participants signed to be admitted to the experiment. Starting from the top, sentence 1) authorises the storage and processing of all data generated by the participant. Sentences 2) and 3) are about the expected behaviour during and after the experiment, namely that the participant will be respectful to the others and will not

disclose the brainstorming questions and ideas generated by the group during the days scheduled for the experimental trials. Sentences a) and b) inform about the rights to privacy and to quit the experiment at any time.

B.2 Entrance Questionnaire

Figure B.2 shows the entrance questionnaire that was filled out by each participant after s/he accepted and signed the consent form. The purpose of this questionnaire was to characterise the participants in terms of sex and age, occupation (mostly students), familiarity with the other members of the group (almost all participants have worked or knew at least one other person on the group), and typing skill (almost everyone reported having a good typing skill). The detailed results from this questionnaire are given in Section 8.3.1 on page 142.

B.3 Brainstorming Instructions

Before the start of the brainstorming sessions, more precisely, before the practise session, I handed in written instructions about the rules of brainstorming to the participants (I also read them out loud). These instructions are depicted in Figure B.3 and are as follows: 1) accept all ideas from the group; 2) new ideas can be based upon existing ideas; 3) ideas should be concise and terse; 4) write ideas as fast as possible; and 5) be silent.

The instructions in Figure B.3 also explain the elements of the user interface of the electronic brainstorming tool that participants would use for the group sessions. This tool, ABTool, is characterised in Section 8.2.4 on page 139. The instructions state that the large shared workspace in the middle region of the window shows group ideas; on top of it is the question that triggers the brainstorming session, and at the bottom is the text field where the participant can write his/her current idea. The right-most region is dedicated to the session status: if it is running or has stopped, how many minutes to the end of the session, the number of people on the group, and how many ideas have been put forward so far.

Departamento de Informática — FCUL
Contacto: António Ferreira, asfe@di.fc.ul.pt
Outubro de 2007

Documento de Autorização

Para a experiência de brainstorming

Eu, abaixo-assinado:

1) Autorizo o uso e tratamento dos dados por mim gerados nesta experiência pelas pessoas responsáveis pela mesma;
2) Comprometo-me a ter uma postura respeitadora relativamente às pessoas envolvidas e à experiência em si;
3) Comprometo-me a não divulgar o conteúdo específico da experiência (questões de brainstorming, ideias produzidas pelo grupo) entre 8 e 19 de Outubro.

Declaro ter conhecimento dos meus direitos:

a) À privacidade;
b) De poder desistir livremente da experiência.

_____

(o voluntário)

Figure B.1: Consent form (in Portuguese).

Departamento de Informática — FCUL
Contacto: António Ferreira, asfe@di.fc.ul.pt
Outubro de 2007

Questionário Inicial

Sexo:

- ☐ Masculino
- ☐ Feminino

Idade: _____

Profissão:

- ☐ Estudante do ____º ano
- ☐ Outra _____

Relacionamento com os outros participantes nesta sessão de brainstorming:

- ☐ Não conheço ninguém deste grupo de brainstorming.
- ☐ Conheço alguém mas não costumo trabalhar em grupo com essa(s) pessoa(s). Quantas? ____
- ☐ Costumo trabalhar em grupo com outras pessoas deste grupo de brainstorming. Quantas?_____

Qual a sua experiência de utilização do teclado do computador?

- ☐ Muita
- ☐ Razoável
- ☐ Pouca

Figure B.2: Entrance questionnaire (in Portuguese).

Departamento de Informática — FCUL
Contacto: António Ferreira, asfe@di.fc.ul.pt
Outubro de 2007

Regras das Sessões de Brainstorming

1. Aceitar todas as ideias do grupo;
2. Ideias podem ser baseadas nas já existentes;
3. Escrever ideias de forma sucinta;
4. Gerar ideias o mais rapidamente possível;
5. Permanecer em silêncio.

Instruções do Software de Brainstorming

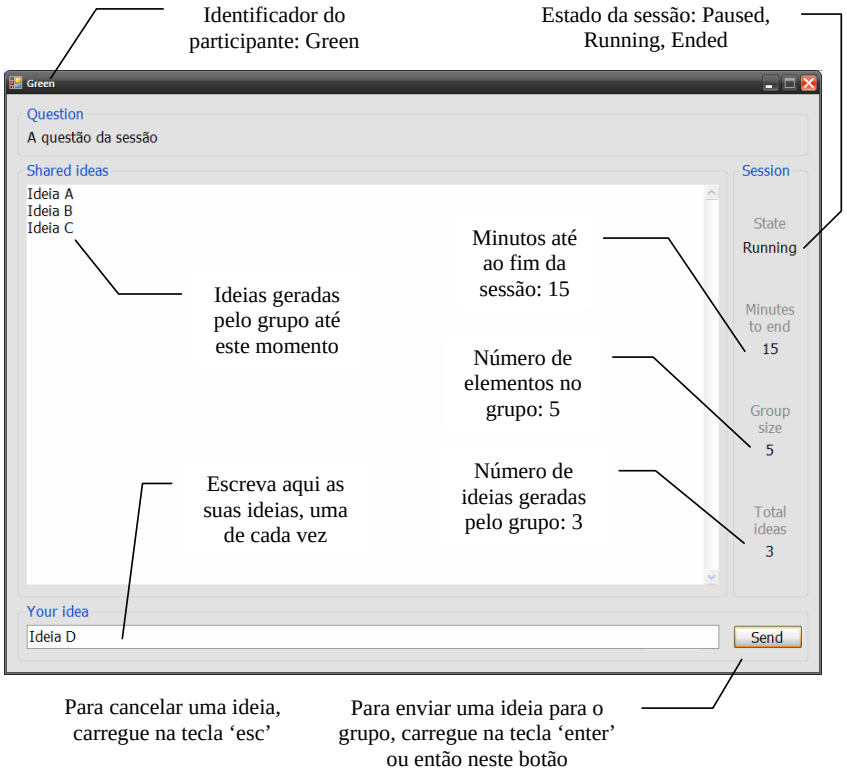


Figure B.3: Brainstorming instructions (in Portuguese).

References

Robert P. Abelson. *Statistics as principled argument*. Lawrence Erlbaum Associates, Hillsdale, NJ, USA, 1995. ISBN 0-8058-0528-1.

Piotr D. Adamczyk and Brian P. Bailey. If not now, when? the effects of interruption at different moments within task execution. In *CHI'04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 271–278, Vienna, Austria, April 2004. ACM Press. ISBN 1-58113-702-8.

John Anderson. *Cognitive psychology and its implications*. Worth Publishers, New York, NY, USA, sixth edition, 2005. ISBN 0-7167-0110-3.

Maria Aneiros, Vladimir Estivill-Castro, and Chengzheng Sun. Group unified histories: An instrument for productive unconstrained co-browsing. In *GROUP'03: Proceedings of the 2003 international ACM SIGGROUP conference on Supporting group work*, pages 330–338, Sanibel Island, FL, USA, November 2003. ACM Press. ISBN 1-58113-693-5.

Pedro Antunes and Carlos J. Costa. Perceived value: A low-cost approach to evaluate meetingware. In *CRIWG'03: Proceedings of the ninth international workshop on Groupware*, pages 109–125, Autrans, France, September 2003. Springer. ISBN 3-540-20117-3.

Pedro Antunes and Tânia Ho. The design of a GDSS meeting preparation tool. *Group Decision and Negotiation*, 10(1):5–25, January 2001. ISSN 0926-2644.

Pedro Antunes, Marcos R. Borges, José A. Pino, and Luis Carriço. Analyzing groupware design by means of usability results. In *CSCWID'05:*

Proceedings of the ninth international conference on Computer supported cooperative work in design, pages 283–288, Conventry, UK, May 2005. IEEE Press. ISBN 1-84600-002-5.

Pedro Antunes, António Ferreira, and José A. Pino. Analyzing shared workspaces design with human-performance models. In *CRIWG'06: Proceedings of the twelfth international workshop on Groupware*, pages 62–77, Medina del Campo, Spain, September 2006a. Springer. ISBN 3-540-39591-1.

Pedro Antunes, João Ramires, and Ana Respício. Addressing the conflicting dimension of groupware: A case study in software requirements validation. *Computing and Informatics*, 25:1001–1024, 2006b. ISSN 1335-9150.

Renata M. Araujo, Flávia M. Santoro, and Marcos R. Borges. A conceptual framework for designing and conducting groupware evaluations. *International Journal of Computer Applications in Technology*, 19(3-4):139–150, May 2004. ISSN 0952-8091.

Kregg Aytes. Comparing collaborative drawing tools and whiteboards: An analysis of the group process. *Computer Supported Cooperative Work*, 4 (1):51–71, March 1995. ISSN 0925-9724.

Ronald M. Baecker, Dimitrios Nastos, Ilona R. Posner, and Kelly L. Mawby. The user-centred iterative design of collaborative writing software. In *CHI'93: Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*, pages 399–405, Amsterdam, Netherlands, April 1993. ACM Press. ISBN 0-89791-575-5.

Brian P. Bailey and Shamsi T. Iqbal. Understanding changes in mental workload during execution of goal-directed tasks and its application for interruption management. *ACM Transactions on Computer-Human Interaction*, 14(4), January 2008. ISSN 1073-0516.

Brian P. Bailey and Joseph A. Konstan. On the need for attention-aware systems: Measuring effects of interruption on task performance, error rate, and affective state. *Computers in Human Behavior*, 22(4):685–708, July 2006. ISSN 0747-5632.

Kevin Baker, Saul Greenberg, and Carl Gutwin. Heuristic evaluation of groupware based on the mechanics of collaboration. In *EHCI'01: Proceedings of the eight IFIP working conference on Engineering for human-computer interaction*, pages 123–138, Toronto, Canada, May 2001. Springer. ISBN 3-540-43044-X.

Kevin Baker, Saul Greenberg, and Carl Gutwin. Empirical development of a heuristic evaluation methodology for shared workspace groupware. In *CSCW'02: Proceedings of the 2002 ACM conference on Computer-supported cooperative work*, pages 96–105, New Orleans, LA, USA, November 2002. ACM Press. ISBN 1-58113-560-2.

Aruna D. Balakrishnan, Susan R. Fussell, and Sara Kiesler. Do visualizations improve synchronous remote collaboration? In *CHI'08: Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1227–1236, Florence, Italy, April 2008. ACM Press. ISBN 978-1-60558-011-1.

Patrick Baudisch, Doug DeCarlo, Andrew T. Duchowski, and Wilson S. Geisler. Focusing on the essential: Considering attention in display design. *Communications of the ACM*, 46(3):60–66, March 2003. ISSN 0001-0782.

Lynn K. Baumeister, Bonnie E. John, and Michael D. Byrne. A comparison of tools for building GOMS models. In *CHI'00: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 502–509, The Hague, Netherlands, April 2000. ACM Press. ISBN 1-58113-216-6.

David V. Beard, Scott Entrikin, Pat Conroy, Nathan C. Wingert, Corey D. Schou, Dana K. Smith, and Kevin M. Denelsbeck. Quick GOMS: A visual software engineering tool for simple rapid time-motion modeling. *Interactions*, 4(3):31–36, May 1997. ISSN 1072-5520.

James Begole, Nicholas E. Matsakis, and John C. Tang. Lilsys: Sensing unavailability. In *CSCW'04: Proceedings of the 2004 ACM conference on Computer-supported cooperative work*, pages 511–514, Chicago, IL, USA, November 2004. ACM Press. ISBN 1-58113-810-5.

Gro Bjerknes and Tone Bratteteig. The memoirs of two survivors: Evaluation of a computer system for cooperative work. In *CSCW'88: Proceedings of the 1988 ACM conference on Computer-supported cooperative work*, pages 167–177, Portland, OR, USA, September 1988. ACM Press. ISBN 0-89791-282-9.

Steve Blythin, John A. Hughes, Steinar Kristoffersen, Tom Rodden, and Mark Rouncefield. Recognising 'success' and 'failure': Evaluating groupware in a commercial context. In *GROUP'97: Proceedings of the international ACM SIGGROUP conference on Supporting group work*, pages 39–46, Phoenix, AZ, USA, November 1997. ACM Press. ISBN 0-89791-897-5.

Barry Boehm, Paul Grunbacher, and Robert O. Briggs. Developing groupware for requirements negotiation: Lessons learned. *IEEE Software*, 18 (3):46–55, 2001. ISSN 0740-7459.

Roger E. Bohn and James E. Short. How much information? 2009: Report on American consumers. Technical report, Global Information Industry Center, University of California, San Diego, CA, USA, December 2009. URL `http://hmi.ucsd.edu`. Retrieved January 2010.

Utpal Bose and David B. Paradice. The effects of integrating cognitive feedback and multi-attribute utility-based multicriteria decision-making methods in GDSS. *Group Decision and Negotiation*, 8(2):157–182, March 1999. ISSN 0926-2644.

Claus Bossen. Representations at work: a national standard for electronic health records. In *CSCW'06: Proceedings of the 2006 ACM conference on Computer-supported cooperative work*, pages 69–78, Banff, Canada, November 2006. ACM Press. ISBN 1-59593-249-6.

John Bowers, Graham Button, and Wes Sharrock. Workflow from within and without: Technology and cooperative work on the print industry shopfloor. In *ECSCW'95: Proceedings of the fourth european conference on Computer-supported cooperative work*, pages 51–66, Stockholm, Sweden, September 1995. Kluwer. ISBN 0-7923-3697-6.

Frederick P. Brooks. *The mythical man-month.* Addison Wesley, Boston, MA, USA, anniversary edition, 1995. ISBN 0-201-83595-9.

Stuart K. Card, Thomas P. Moran, and Allen Newell. The keystroke-level model for user performance time with interactive systems. *Communications of the ACM*, 23(7):396–410, July 1980. ISSN 0001-0782.

Stuart K. Card, Allen Newell, and Thomas P. Moran. *The psychology of human-computer interaction*. Lawrence Erlbaum Associates, Mahwah, NJ, USA, 1983. ISBN 0-89859-859-1.

Peter H. Carstensen and Morten Nielsen. Characterizing modes of coordination. In *GROUP'01: Proceedings of the 2001 international ACM SIGGROUP conference on Supporting group work*, pages 81–90, Boulder, CO, USA, September 2001. ACM Press. ISBN 1-58113-294-8.

Daniel Chen and Roel Vertegaal. Using mental load for managing interruptions in physiologically attentive user interfaces. In *CHI'04: Extended abstracts on Human factors in computing systems*, pages 1513–1516, Vienna, Austria, April 2004. ACM Press. ISBN 1-58113-703-6.

Ed H. Chi. The social web: Research and opportunities. *Computer*, 41(9): 88–91, September 2008. ISSN 0018-9162.

Andy Cockburn and Tony Dale. CEVA: A tool for collaborative video analysis. In *GROUP'97: Proceedings of the international ACM SIGGROUP conference on Supporting group work*, pages 47–55, Phoenix, AZ, USA, November 1997. ACM Press. ISBN 0-89791-897-5.

Gilbert Cockton and Alan Woolrych. Sale must end: Should discount methods be cleared off HCI's shelves? *Interactions*, 9(5):13–18, September 2002. ISSN 1072-5520.

Joanie B. Connell, Gerald A. Mendelsohn, Richard W. Robins, and John Canny. Effects of communication medium on interpersonal perceptions: Don't hang up on the telephone yet. In *GROUP'01: Proceedings of the 2001 international ACM SIGGROUP conference on Supporting group work*, pages 117–124, Boulder, CO, USA, September 2001. ACM Press. ISBN 1-58113-294-8.

Terry Connolly, Leonard M. Jessup, and Joseph S. Valacich. Effects of anonymity and evaluative tone on idea generation in computer-mediated groups. *Management Science*, 36(6):689–703, June 1990. ISSN 0025-1909.

Andrew R. Conway, Nelson Cowan, and Michael F. Bunting. The cocktail party phenomenon revisited: The importance of working memory capacity. *Psychonomic Bulletin and Review*, 8(2):331–335, June 2001. ISSN 1531-5320.

Colleen Cool, Robert S. Fish, Robert E. Kraut, and C.M. Lowery. Iterative design of video communication systems. In *CSCW'92: Proceedings of the 1992 ACM conference on Computer-supported cooperative work*, pages 25–32, Toronto, Canada, November 1992. ACM Press. ISBN 0-89791-542-9.

Andy Crabtree, Tom Rodden, Peter Tolmie, and Graham Button. Ethnography considered harmful. In *CHI'09: Proceedings of the twenty-seventh international conference on Human factors in computing systems*, pages 879–888, Boston, MA, USA, April 2009. ACM Press. ISBN 978-1-60558-246-7.

Terrence Crowley, Paul Milazzo, Ellie Baker, Harry Forsdick, and Raymond Tomlinson. MMConf: An infrastructure for building shared multimedia applications. In *CSCW'90: Proceedings of the 1990 ACM conference on Computer-supported cooperative work*, pages 329–342, Los Angeles, CA, USA, November 1990. ACM Press. ISBN 0-89791-402-3.

Edward Cutrell, Mary Czerwinski, and Eric Horvitz. Notification, disruption, and memory: Effects of messaging interruptions on memory and performance. In *Interact'01: Proceedings of the eight IFIP international conference on Human-Computer Interaction*, pages 263–269, Tokyo, Japan, July 2001. IOS Press. ISBN 1-58603-188-0.

Mary Czerwinski, Edward Cutrell, and Eric Horvitz. Instant messaging: Effects of relevance and timing. In *HCI'00: Proceedings of the HCI conference*, pages 71–76, Sunderland, UK, September 2000. Springer. ISBN 1-85233-318-9.

Mary Czerwinski, Eric Horvitz, and Susan Wilhite. A diary study of task switching and interruptions. In *CHI'04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 175–182, Vienna, Austria, April 2004. ACM Press. ISBN 1-58113-702-8.

Laurie Damianos, Lynette Hirschman, Robyn Kozierok, Jeffrey Kurtz, Andrew Greenberg, Kimberley Walls, Sharon Laskowski, and Jean Scholtz. Evaluation for collaborative systems. *ACM Computing Surveys*, 31(2), June 1999. ISSN 0360-0300.

Ann Davey and David Olson. Multiple criteria decision making models in group decision support. *Group Decision and Negotiation*, 7(1):55–75, January 1998. ISSN 0926-2644.

Alan R. Dennis, Joseph S. Valacich, and Jay F. Nunamaker. Group, subgroup, and nominal group idea generation: New rules for a new media? *Journal of Management*, 20(4):723–736, 1994. ISSN 0149-2063.

Alan R. Dennis, Joseph S. Valacich, Terry Connolly, and Bayard E. Wynne. Process structuring in electronic brainstorming. *Information Systems Research*, 7(2):268–277, June 1996. ISSN 1047-7047.

Prasun Dewan and Rajiv Choudhary. Coupling the user interfaces of a multiuser program. *ACM Transactions on Computer-Human Interaction*, 2(1):1–39, March 1995. ISSN 1073-0516.

Alan Dix, Janet Finlay, Gregory D. Abowd, and Russell Beale. *Human-computer interaction*. Prentice Hall, Essex, UK, third edition, 2003. ISBN 0-13-046109-1.

Sarah A. Douglas and Arthur E. Kirkpatrick. Model and representation: The effect of visual feedback on human performance in a color picker interface. *ACM Transactions on Graphics*, 18(2):96–127, 1999. ISSN 0730-0301.

Paul Dourish. Implications for design. In *CHI'06: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 541–550, Montreal, Canada, April 2006. ACM Press. ISBN 1-59593-178-3.

Paul Dourish and Victoria Bellotti. Awareness and coordination in shared workspaces. In *CSCW'92: Proceedings of the 1992 ACM conference on Computer-supported cooperative work*, pages 107–114, Toronto, Canada, November 1992. ACM Press. ISBN 0-89791-542-9.

Paul Dourish and Sara Bly. Portholes: Supporting awareness in a distributed work group. In *CHI'92: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 541–547, Monterey, CA, USA, May 1992. ACM Press. ISBN 0-89791-513-5.

Richard V. Dragan. Collaborating with Office 2003. *PC Magazine*, September 2003. URL `http://www.pcmag.com/article2/0,2817,1273092,00.asp`. Retrieved November 2008.

Jill Drury and Marian G. Williams. A framework for role-based specification and evaluation of awareness support in synchronous collaborative applications. In *WETICE'02: Proceedings of the eleventh IEEE international workshops on Enabling technologies*, pages 12–17, Pittsburgh, PA, USA, June 2002. IEEE Press. ISBN 0-7695-1748-X.

Jill L. Drury, Jean Scholtz, and David Kieras. Adapting GOMS to model human-robot interaction. In *HRI'07: Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 41–48, Arlington, VA, USA, March 2007. ACM Press. ISBN 978-1-59593-617-2.

Honglu Du, Mary B. Rosson, John Carroll, and Craig Ganoe. I felt like a contributing member of the class: Increasing class participation with ClassCommons. In *GROUP'09: Proceedings of the ACM 2009 international conference on Supporting group work*, pages 233–242, Sanibel Island, FL, USA, May 2009. ACM Press. ISBN 978-1-60558-500-0.

Shelli Dubs and Stephen C. Hayne. Distributed facilitation: A concept whose time has come? In *CSCW'92: Proceedings of the 1992 ACM conference on Computer-supported cooperative work*, pages 314–321, Toronto, Canada, November 1992. ACM Press. ISBN 0-89791-542-9.

Paula Durlach. Change blindness and its implications for complex monitoring and control systems design and operator training. *Human-Computer Interaction*, 19(4):423–451, 2004. ISSN 0737-0024.

Jeff Dyck, Carl Gutwin, Sriram Subramanian, and Christopher Fedak. High-performance telepointers. In *CSCW'04: Proceedings of the 2004 ACM conference on Computer-supported cooperative work*, pages 172–181, Chicago, IL, USA, November 2004. ACM Press. ISBN 1-58113-810-5.

Clarence Ellis and Jacques Wainer. A conceptual model of groupware. In *CSCW'94: Proceedings of the 1994 ACM conference on Computer-supported cooperative work*, pages 79–88, Chapel Hill, NC, USA, October 1994. ACM Press. ISBN 0-89791-689-1.

Clarence A. Ellis, Simon J. Gibbs, and Gail Rein. Groupware: Some issues and experiences. *Communications of the ACM*, 34(1):39–58, 1991. ISSN 0001-0782.

Martin Eppler and Jeanne Mengis. The concept of information overload: A review of literature from organization science, accounting, marketing, MIS, and related disciplines. *The Information Society*, 20(5):325–344, 2004. ISSN 0197-2243.

Michael W. Eysenck and Mark T. Keane. *Cognitive psychology: A student's handbook*. Psychology Press, East Sussex, UK, fourth edition, 2000. ISBN 0-86377-550-0.

António Ferreira and Pedro Antunes. Quantitative evaluation of workspace collaboration. In *CSCWID'06: Proceedings of the tenth international conference on Computer supported cooperative work in design*, pages 1065–1070, Nanjing, China, May 2006a. IEEE Press. ISBN 1-4244-0164-X.

António Ferreira and Pedro Antunes. Dispositivos de gestão da atenção em sistemas colaborativos. In *Interacção'06: Actas da segunda conferência nacional em Interacção pessoa-máquina*, pages 57–60, Minho, Portugal, October 2006b. Grupo Português Computação Gráfica. ISBN 972-98464-7-2.

António Ferreira and Pedro Antunes. On the need for a framework for attentive groupware systems. In *SociUM'07: Proceedings of the first workshop on Adaptation and personalisation in social systems*, pages 5–15, Corfu, Greece, June 2007a.

António Ferreira and Pedro Antunes. A technique for evaluating shared workspaces efficiency. In Weiming Shen, editor, *CSCW in design III*, pages 82–91. Springer, Berlin, Germany, 2007b. ISBN 978-3-540-72862-7.

António Ferreira and Pedro Antunes. Tackling information overload in electronic brainstorming. In *GDN'08: Proceedings of the ninth Group decision and negotiation meeting*, pages 83–89, Coimbra, Portugal, June 2008a. Instituto de Engenharia de Sistemas e Computadores de Coimbra.

António Ferreira and Pedro Antunes. An attentive groupware device to mitigate information overload. In *DSV-IS'08: Proceedings of the fifteenth international workshop on Design, specification, and validation of interactive systems*, pages 29–42, Kingston, Canada, July 2008b. Springer. ISBN 978-3-540-70568-0.

António Ferreira, Valeria Herskovic, and Pedro Antunes. Attention-based management of information flows in synchronous electronic brainstorming. In *CRIWG'08: Proceedings of the fourteenth international workshop on Groupware*, pages 1–16, Omaha, NE, USA, September 2008. Springer. ISBN 978-3-540-92830-0.

António Ferreira, Pedro Antunes, and José A. Pino. Evaluating shared workspace performance using human information processing models. *Information Research*, 14(1), March 2009. ISSN 1368-1613. URL `http://informationr.net/ir/14-1/paper388.html`.

António Ferreira, Pedro Antunes, and Valeria Herskovic. Improving group attention: An experiment with synchronous brainstorming. *Group Decision and Negotiation*, 2010. ISSN 0926-2644. To appear.

Graeme E. Field. Experimentus interruptus. *SIGCHI Bulletin*, 19(2):42–46, 1987. ISSN 0736-6906.

Jerry Fjermestad and Starr Hiltz. An assessment of group support systems experimental research: Methodology and results. *Journal of Management Information Systems*, 15(3):7–149, 1999. ISSN 0742-1222.

James Fogarty and Scott E. Hudson. Toolkit support for developing and deploying sensor-based statistical models of human situations. In *CHI'07:*

Proceedings of the SIGCHI conference on Human factors in computing systems, pages 135–144, San Jose, CA, USA, April 2007. ACM Press. ISBN 978-1-59593-593-9.

James Fogarty, Scott E. Hudson, Christopher G. Atkeson, Daniel Avrahami, Jodi Forlizzi, Sara Kiesler, Johnny C. Lee, and Jie Yang. Predicting human interruptibility with sensors. *ACM Transactions on Computer-Human Interaction*, 12(1):119–146, March 2005a. ISSN 1073-0516.

James Fogarty, Andrew J. Ko, Htet Htet Aung, Elspeth Golden, Karen P. Tang, and Scott E. Hudson. Examining task engagement in sensor-based statistical models of human interruptibility. In *CHI'05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 331–340, Portland, OR, USA, April 2005b. ACM Press. ISBN 1-58113-998-5.

Clifton Forlines, Chia Shen, Daniel Wigdor, and Ravin Balakrishnan. Exploring the effects of group size and display configuration on visual search. In *CSCW'06: Proceedings of the 2006 ACM conference on Computer-supported cooperative work*, pages 11–20, Banff, Canada, November 2006. ACM Press. ISBN 1-59593-249-6.

Mike Fraser, Michael R. McCarthy, Muneeb Shaukat, and Phillip Smith. Seconds matter: Improving distributed coordination by tracking and visualizing display trajectories. In *CHI'07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1303–1312, San Jose, CA, USA, April 2007. ACM Press. ISBN 978-1-59593-593-9.

Susan R. Fussell, Robert E. Kraut, and Jane Siegel. Coordination of communication: Effects of shared visual context on collaborative work. In *CSCW'00: Proceedings of the 2000 ACM conference on Computer-supported cooperative work*, pages 21–30, Philadelphia, PA, USA, December 2000. ACM Press. ISBN 1-58113-222-0.

Susan R. Fussell, Leslie D. Setlock, and Robert E. Kraut. Effects of head-mounted and scene-oriented video systems on remote collaboration on physical tasks. In *CHI'03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 513–520, Ft. Lauderdale, FL, USA, April 2003. ACM Press. ISBN 1-58113-630-7.

Stephen Gale. Adding audio and video to an office environment. In John M. Bowers and Steve Benford, editors, *Studies in computer supported cooperative work: Theory, practice and design*, pages 49–62. North-Holland, Amsterdam, Netherlands, 1991. ISBN 0-444-88811-X.

John F. Gantz. The diverse and exploding digital universe. Technical report, International Data Corporation, Framingham, MA, USA, March 2008. URL `http://www.emc.com/collateral/analyst-reports/diverse-exploding-digital-universe.pdf`. Retrieved January 2010.

Maia Garau, Mel Slater, Simon Bee, and Martina A. Sasse. The impact of eye gaze on communication using humanoid avatars. In *CHI'01: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 309–316, Seattle, WA, USA, March 2001. ACM Press. ISBN 1-58113-327-8.

Samuel J. Gibbs. LIZA: An extensible groupware toolkit. *SIGCHI Bulletin*, 20(SI):29–35, March 1989. ISSN 0736-6906.

Jennifer Gluck, Andrea Bunt, and Joanna McGrenere. Matching attentional draw with utility in interruption. In *CHI'07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 41–50, San Jose, CA, USA, April 2007. ACM Press. ISBN 978-1-59593-593-9.

Wayne D. Gray and Deborah A. Boehm-Davis. Milliseconds matter: An introduction to microstrategies and to their use in describing and predicting interactive behavior. *Journal of Experimental Psychology: Applied*, 6(4): 322–335, 2000. ISSN 1076-898X.

Wayne D. Gray, Bonnie E. John, and Michael E. Atwood. Project Ernestine: Validating a GOMS analysis for predicting and explaining real-world task performance. *Human-Computer Interaction*, 8(3):237–309, 1993. ISSN 0737-0024.

Saul Greenberg and Bill Buxton. Usability evaluation considered harmful (some of the time). In *CHI'08: Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 111–120, Florence, Italy, April 2008. ACM Press. ISBN 978-1-60558-011-1.

Saul Greenberg and Mark Roseman. Using a room metaphor to ease transitions in groupware. In Mark S. Ackerman, Volkmar Pipek, and Volker Wulf, editors, *Sharing experience: Beyond knowledge management*. MIT Press, Cambridge, MA, USA, 2003. ISBN 0-262-01195-6.

Mary-Liz Grisé and R. Brent Gallupe. Information overload: Addressing the productivity paradox in face-to-face electronic meetings. *Journal of Management Information Systems*, 16(3):157–185, December 1999. ISSN 0742-1222.

Jonathan Grudin. Why CSCW applications fail: Problems in the design and evaluation of organizational interfaces. In *CSCW'88: Proceedings of the 1988 ACM conference on Computer-supported cooperative work*, pages 85–93, Portland, OR, USA, September 1988. ACM Press. ISBN 0-89791-282-9.

Jonathan Grudin. Groupware and social dynamics: Eight challenges for developers. *Communications of the ACM*, 37(1):92–105, January 1994. ISSN 0001-0782.

Daniel Gruen, Steven L. Rohall, Suzanne Minassian, Bernard Kerr, Paul Moody, Bob Stachel, Martin Wattenberg, and Eric Wilcox. Lessons from the reMail prototypes. In *CSCW'04: Proceedings of the 2004 ACM conference on Computer-supported cooperative work*, pages 152–161, Chicago, IL, USA, November 2004. ACM Press. ISBN 1-58113-810-5.

Carl Gutwin and Saul Greenberg. Design for individuals, design for groups: Tradeoffs between power and workspace awareness. In *CSCW'98: Proceedings of the 1998 ACM conference on Computer-supported cooperative work*, pages 207–216, Seattle, WA, USA, November 1998. ACM Press. ISBN 1-58113-009-0.

Carl Gutwin and Saul Greenberg. The effects of workspace awareness support on the usability of real-time distributed groupware. *ACM Transactions on Computer-Human Interaction*, 6(3):243–281, September 1999. ISSN 1073-0516.

Carl Gutwin and Saul Greenberg. The mechanics of collaboration: Developing low cost usability evaluation methods for shared workspaces. In *WETICE'00: Proceedings of the ninth IEEE international workshops on Enabling technologies*, pages 98–103, Gaithersburg, MD, USA, June 2000. IEEE Press. ISBN 0-7695-0798-0.

Carl Gutwin and Saul Greenberg. A descriptive framework of workspace awareness for real-time groupware. *Computer Supported Cooperative Work*, 11(3):411–446, September 2002. ISSN 0925-9724.

Carl Gutwin and Reagan Penner. Improving interpretation of remote gestures with telepointer traces. In *CSCW'02: Proceedings of the 2002 ACM conference on Computer-supported cooperative work*, pages 49–57, New Orleans, LA, USA, November 2002. ACM Press. ISBN 1-58113-560-2.

Carl Gutwin, Jeff Dyck, and Jennifer Burkitt. Using cursor prediction to smooth telepointer jitter. In *GROUP'03: Proceedings of the 2003 international ACM SIGGROUP conference on Supporting group work*, pages 294–301, Sanibel Island, FL, USA, November 2003. ACM Press. ISBN 1-58113-693-5.

Carl Gutwin, Steve Benford, Jeff Dyck, Mike Fraser, Ivan Vaghi, and Chris Greenhalgh. Revealing delay in collaborative environments. In *CHI'04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 503–510, Vienna, Austria, April 2004. ACM Press. ISBN 1-58113-702-8.

Stephen Haag, M.K. Raja, and L.L. Schkade. Quality function deployment usage in software development. *Communications of the ACM*, 39(1):41–49, January 1996. ISSN 0001-0782.

Mark Hackman. Microsoft to take Office to the Web, finally. *PC Magazine*, October 2008. URL `http://www.pcmag.com/article2/0,2817,2333441,00.asp`. Retrieved November 2008.

Richard R. Harper, J.A. Hughes, and Dan Z. Shapiro. Working in harmony: An examination of computer technology in air traffic control. In

ECSCW'89: Proceedings of the first european conference on Computer-supported cooperative work, pages 73–87, Gatwick, UK, September 1989.

Jörg Hauber, Holger Regenbrecht, Mark Billinghurst, and Andy Cockburn. Spatiality in videoconferencing: Trade-offs between efficiency and social presence. In *CSCW'06: Proceedings of the 2006 ACM conference on Computer-supported cooperative work*, pages 413–422, Banff, Canada, November 2006. ACM Press. ISBN 1-59593-249-6.

Steven R. Haynes, Sandeep Purao, and Amie L. Skattebo. Situating evaluation in scenarios of use. In *CSCW'04: Proceedings of the 2004 ACM conference on Computer-supported cooperative work*, pages 92–101, Chicago, IL, USA, November 2004. ACM Press. ISBN 1-58113-810-5.

Christian Heath and Paul Luff. Documents and professional practise: 'bad' organisational reasons for 'good' clinical records. In *CSCW'96: Proceedings of the 1996 ACM conference on Computer-supported cooperative work*, pages 354–363, Boston, MA, USA, November 1996. ACM Press. ISBN 0-89791-765-0.

William G. Heninger, Alan R. Dennis, and Kelly M. Hilmer. Individual cognition and dual-task interference in group support systems. *Information Systems Research*, 17(4):415–424, December 2006. ISSN 1047-7047.

James D. Herbsleb, David L. Atkins, David G. Boyer, Mark Handel, and Thomas A. Finholt. Introducing instant messaging and chat in the workplace. In *CHI'02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 171–178, Minneapolis, MN, USA, April 2002. ACM Press. ISBN 1-58113-453-3.

Valeria Herskovic, José A. Pino, Sergio F. Ochoa, and Pedro Antunes. Evaluation methods for groupware systems. In *CRIWG'07: Proceedings of the thirtieth international workshop on Groupware*, pages 328–336, Bariloche, Argentina, September 2007. Springer. ISBN 978-3-540-74811-3.

Jason Hill and Carl Gutwin. Awareness support in a groupware widget toolkit. In *GROUP'03: Proceedings of the 2003 international ACM SIGGROUP*

conference on Supporting group work, pages 258–267, Sanibel Island, FL, USA, November 2003. ACM Press. ISBN 1-58113-693-5.

Jason Hill and Carl Gutwin. The MAUI toolkit: Groupware widgets for group awareness. *Computer Supported Cooperative Work*, 13(5):539–571, December 2004. ISSN 0925-9724.

Ralph D. Hill, Tom Brinck, Steven L. Rohall, John F. Patterson, and Wayne Wilner. The Rendezvous architecture and language for constructing multiuser applications. *ACM Transactions on Computer-Human Interaction*, 1(2):81–125, June 1994. ISSN 1073-0516.

Paul Holleis, Friederike Otto, Heinrich Hussmann, and Albrecht Schmidt. Keystroke-level model for advanced mobile phone interaction. In *CHI'07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1505–1514, San Jose, CA, USA, April 2007. ACM Press. ISBN 978-1-59593-593-9.

Eric Horvitz and Johnson Apacible. Learning and reasoning about interruption. In *ICMI'03: Proceedings of the fifth international conference on Multimodal interfaces*, pages 20–27, Vancouver, Canada, November 2003. ACM Press. ISBN 1-58113-621-8.

Eric Horvitz, Carl Kadie, Tim Paek, and David Hovel. Models of attention in computing and communication: From principle to applications. *Communications of the ACM*, 46(3):52–59, March 2003. ISSN 0001-0782.

David C. Howell. *Statistical methods for psychology*. Wadsworth, Belmont, CA, USA, sixth edition, 2007. ISBN 0-495-09361-0.

Scott E. Hudson, Bonnie E. John, Keith Knudsen, and Michael D. Byrne. A tool for creating predictive performance models from user interface demonstrations. In *UIST'99: Proceedings of the twelfth annual ACM symposium on User interface software and technology*, pages 93–102, Asheville, NC, USA, November 1999. ACM Press. ISBN 1-58113-075-9.

John Hughes, Val King, Tom Rodden, and Hans Andersen. Moving out from the control room: Ethnography in system design. In *CSCW'94: Proceedings of the 1994 ACM conference on Computer-supported cooperative*

work, pages 429–439, Chapel Hill, NC, USA, October 1994. ACM Press. ISBN 0-89791-689-1.

Charles M. Hymes and Gary M. Olson. Unblocking brainstorming through the use of a simple group editor. In *CSCW'92: Proceedings of the 1992 ACM conference on Computer-supported cooperative work*, pages 99–106, Toronto, Canada, November 1992. ACM Press. ISBN 0-89791-542-9.

Aulikki Hyrskykari, Päivi Majaranta, Antti Aaltonen, and Kari-Jouko Räihä. Design issues of iDICT: A gaze-assisted translation aid. In *ETRA'00: Proceedings of the 2000 symposium on Eye tracking research and applications*, pages 9–14, Palm Beach Gardens, FL, USA, November 2000. ACM Press. ISBN 1-58113-280-8.

Yusuke Ichikawa, Ken ichi Okada, Giseok Jeong, Shunsuke Tanaka, and Yutaka Matsushita. MAJIC videoconferencing system: Experiments, evaluation and improvement. In *ECSCW'95: Proceedings of the fourth european conference on Computer-supported cooperative work*, pages 279–292, Stockholm, Sweden, September 1995. Kluwer. ISBN 0-7923-3697-6.

Claudia-Lavinia Ignat and Moira C. Norrie. Draw-together: Graphical editor for collaborative drawing. In *CSCW'06: Proceedings of the 2006 ACM conference on Computer-supported cooperative work*, pages 269–278, Banff, Canada, November 2006. ACM Press. ISBN 1-59593-249-6.

Shamsi T. Iqbal and Brian P. Bailey. Investigating the effectiveness of mental workload as a predictor of opportune moments for interruption. In *CHI'05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1489–1492, Portland, OR, USA, April 2005. ACM Press. ISBN 1-58113-998-5.

Shamsi T. Iqbal and Brian P. Bailey. Leveraging characteristics of task structure to predict the cost of interruption. In *CHI'06: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 741–750, Montreal, Canada, April 2006. ACM Press. ISBN 1-59593-178-3.

Shamsi T. Iqbal and Brian P. Bailey. Effects of intelligent notification management on users and their tasks. In *CHI'08: Proceedings of the*

twenty-sixth annual SIGCHI conference on Human factors in computing systems, pages 93–102, Florence, Italy, April 2008. ACM Press. ISBN 978-1-60558-011-1.

Hiroshi Ishii, Minoru Kobayashi, and Jonathan Grudin. Integration of interpersonal space and shared workspace: ClearBoard design and experiments. *ACM Transactions on Information Systems*, 11(4):349–375, October 1993. ISSN 1046-8188.

Melody Y. Ivory and Marti A. Hearst. The state of the art in automating usability evaluation of user interfaces. *ACM Computing Surveys*, 33(4): 470–516, December 2001. ISSN 0360-0300.

Michal Jacovi, Vladimir Soroka, Gail Gilboa-Freedman, Sigalit Ur, Elad Shahar, and Natalia Marmasse. The chasms of CSCW: A citation graph analysis of the CSCW conference. In *CSCW'06: Proceedings of the 2006 ACM conference on Computer-supported cooperative work*, pages 289–298, Banff, Canada, November 2006. ACM Press. ISBN 1-59593-249-6.

Tracy Jenkin, Jesse McGeachie, David Fono, and Roel Vertegaal. eyeView: Focus+context views for large group video conferences. In *CHI'05: Extended abstracts on Human factors in computing systems*, pages 1497–1500, Portland, OR, USA, April 2005. ACM Press. ISBN 1-59593-002-7.

Bonnie John. Why GOMS? *Interactions*, 2(4):80–89, 1995. ISSN 1072-5520.

Bonnie John, Alonso Vera, Michael Matessa, Michael Freed, and Roger Remington. Automating CPM-GOMS. In *CHI'02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 147–154, Minneapolis, MN, USA, April 2002. ACM Press. ISBN 1-58113-453-3.

Bonnie E. John. Extensions of GOMS analyses to expert performance requiring perception of dynamic visual and auditory information. In *CHI'90: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 107–116, Seattle, WA, USA, April 1990. ACM Press. ISBN 0-201-50932-6.

Bonnie E. John and David E. Kieras. Using GOMS for user interface design and evaluation: Which technique? *ACM Transactions on Computer-Human Interaction*, 3(4):287–319, December 1996a. ISSN 1073-0516.

Bonnie E. John and David E. Kieras. The GOMS family of user interface analysis techniques: Comparison and contrast. *ACM Transactions on Computer-Human Interaction*, 3(4):320–351, December 1996b. ISSN 1073-0516.

Bonnie E. John, Konstantine Prevas, Dario D. Salvucci, and Ken Koedinger. Predictive human performance modeling made easy. In *CHI'04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 455–462, Vienna, Austria, April 2004. ACM Press. ISBN 1-58113-702-8.

Sasa Junuzovic and Prasun Dewan. Lazy scheduling of processing and transmission tasks in collaborative systems. In *GROUP'09: Proceedings of the ACM 2009 international conference on Supporting group work*, pages 159–168, Sanibel Island, FL, USA, May 2009. ACM Press. ISBN 978-1-60558-500-0.

Peter J. Kammer, Gregory A. Bolcer, Richard N. Taylor, Arthur S. Hitomi, and Mark Bergman. Techniques for supporting dynamic and adaptive workflow. *Computer Supported Cooperative Work*, 9(3-4):269–292, August 2000. ISSN 0925-9724.

Setrag Khoshafian and Marek Buckiewicz. *Introduction to groupware, workflow, and workgroup computing.* Wiley, New York, NY, USA, 1995. ISBN 0-471-02946-7.

David Kieras. Towards a practical GOMS model methodology for user interface design. In Martin Helander, editor, *The handbook of human-computer interaction*, pages 135–158. Elsevier, Amsterdam, Netherlands, 1988. ISBN 0-444-88673-7.

David Kieras. GOMS models for task analysis. In Dan Diaper and Neville Stanton, editors, *The handbook of task analysis for human-computer*

interaction, pages 83–116. Lawrence Erlbaum Associates, Mahwah, NJ, USA, 2003. ISBN 0-8058-4432-5.

David E. Kieras and Thomas P. Santoro. Computational GOMS modeling of a complex team task: Lessons learned. In *CHI'04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 97–104, Vienna, Austria, April 2004. ACM Press. ISBN 1-58113-702-8.

David E. Kieras, Scott D. Wood, Kasem Abotel, and Anthony Hornof. GLEAN: A computer-based tool for rapid GOMS model usability evaluation of user interface designs. In *UIST'95: Proceedings of the eight annual ACM symposium on User interface software and technology*, pages 91–100, Pittsburgh, PA, USA, November 1995. ACM Press. ISBN 0-89791-709-X.

Russell Kruger, Sheelagh Carpendale, Stacey D. Scott, and Saul Greenberg. Roles of orientation in tabletop collaboration: Comprehension, coordination and communication. *Computer Supported Cooperative Work*, 13(5-6): 501–537, December 2004. ISSN 0925-9724.

David LaBerge. Attention. In Benjamin M. Bly and David E. Rumelhart, editors, *Cognitive Science*, pages 44–98. Academic Press, San Diego, CA, USA, 1999. ISBN 0-12-601730-1.

John T. Langton, Timothy J. Hickey, and Richard Alterman. Integrating tools and resources: A case study in building educational groupware for collaborative programming. *Journal of Computing Sciences in Colleges*, 19(5):140–153, May 2004. ISSN 1937-4771.

Jure Leskovec and Eric Horvitz. Planetary-scale views on a large instant-messaging network. In *WWW'08: Proceedings of the seventhieth international conference on World Wide Web*, pages 915–924, Beijing, China, April 2008. ACM Press. ISBN 978-1-60558-085-2.

Peter Lyman and Hal R. Varian. How much information? Technical report, School of Information Management and Systems, University of California, Berkeley, CA, USA, 2003. URL `http://www.sims.berkeley.edu/how-much-info-2003`. Retrieved May 2009.

Thomas W. Malone and Kevin Crowston. The interdisciplinary study of coordination. *ACM Computing Surveys*, 26(1):87–119, 1994. ISSN 0360-0300.

Gloria Mark, Jörg M. Haake, and Norbert A. Streitz. Hypermedia use in group work: Changing the product, process, and strategy. *Computer Supported Cooperative Work*, 6(4):327–368, December 1997. ISSN 0925-9724.

Gloria Mark, Victor M. Gonzalez, and Justin Harris. No task left behind? examining the structure of fragmented work. In *CHI'05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 321–330, Portland, OR, USA, April 2005. ACM Press. ISBN 1-58113-998-5.

René Marois and Jason Ivanoff. Capacity limits of information processing in the brain. *Trends in Cognitive Sciences*, 9(6):296–305, 2005. ISSN 1364-6613.

David Martin, Mark Rouncefield, Jacki O'Neill, Mark Hartswood, and Dave Randall. Timing in the art of integration: That's how the Bastille got stormed. In *GROUP'05: Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work*, pages 313–322, Sanibel Island, FL, USA, November 2005. ACM Press. ISBN 1-59593-223-2.

D. Scott McCrickard and Christa M. Chewar. Attuning notification design to user goals and attention costs. *Communications of the ACM*, 46(3): 67–72, March 2003. ISSN 0001-0782.

Susan E. McDaniel, Gary M. Olson, and Joseph C. Magee. Identifying and analyzing multiple threads in computer-mediated and face-to-face conversations. In *CSCW'96: Proceedings of the 1996 ACM conference on Computer-supported cooperative work*, pages 39–47, Boston, MA, USA, November 1996. ACM Press. ISBN 0-89791-765-0.

Daniel C. McFarlane. Comparison of four primary methods for coordinating the interruption of people in human-computer interaction. *Human-Computer Interaction*, 17(1):63–139, 2002. ISSN 0737-0024.

Joseph E. McGrath. *Groups: Interaction and performance*. Prentice Hall, Englewood Cliffs, NJ, USA, 1984. ISBN 0-13-365700-0.

Sean M. McNee, Istvan Albert, Dan Cosley, Prateep Gopalkrishnan, Shyong K. Lam, Al Mamunur Rashid, Joseph A. Konstan, and John Riedl. On the recommending of citations for research papers. In *CSCW'02: Proceedings of the 2002 ACM conference on Computer-supported cooperative work*, pages 116–125, New Orleans, LA, USA, November 2002. ACM Press. ISBN 1-58113-560-2.

Brian E. Mennecke, Joseph S. Valacich, and Bradley C. Wheeler. The effects of media and task on user performance: A test of the task-media fit hypothesis. *Group Decision and Negotiation*, 9(6):507–529, November 2000. ISSN 0926-2644.

Antonios Michailidis and Roy Rada. Comparative study on the effects of groupware and conventional technologies on the efficiency of collaborative writing. *Computer Supported Cooperative Work*, 3(3):327–357, September 1994. ISSN 0925-9724.

Matthew B. Miles and Michael Huberman. *Qualitative data analysis: An expanded sourcebook*. Sage Publications, Thousand Oaks, CA, USA, 1994. ISBN 0-8039-5540-5.

Daihwan Min, Sanghoe Koo, Yun-Hyung Chung, and Bokryeul Kim. Distributed GOMS: An extension of GOMS to group task. In *SMC'99: Proceedings of the IEEE international conference on Systems, man, and cybernetics*, pages 720–725, Tokyo, Japan, October 1999. IEEE Press. ISBN 0-7803-5731-0.

Jane N. Mosier and Susan G. Tammaro. When are group scheduling tools useful? *Computer Supported Cooperative Work*, 6(1):53–70, March 1997. ISSN 0925-9724.

Eduardo Mosqueira-Rey, Julio Rivela-Carballal, and Vicente Moret-Bonillo. Integrating GOMS models and logging methods into a single tool for evaluating the usability of intelligent systems. In *SMC'04: Proceedings of the IEEE international conference on Systems, man, and cybernetics*,

pages 5142–5147, The Hague, Netherlands, October 2004. IEEE Press. ISBN 0-7803-8566-7.

Jonathan P. Munson and Prasun Dewan. A flexible object merging framework. In *CSCW'94: Proceedings of the 1994 ACM conference on Computer-supported cooperative work*, pages 231–242, Chapel Hill, NC, USA, October 1994. ACM Press. ISBN 0-89791-689-1.

Miguel A. Nacenta, Dzmitry Aliakseyeu, Sriram Subramanian, and Carl Gutwin. A comparison of techniques for multi-display reaching. In *CHI'05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 371–380, Portland, OR, USA, April 2005. ACM Press. ISBN 1-58113-998-5.

Murli Nagasundaram and Alan R. Dennis. When a group is not a group: The cognitive foundation of group idea generation. *Small Group Research*, 24(4):463–489, November 1993. ISSN 1046-4964.

Dennis C. Neale, John M. Carroll, and Mary B. Rosson. Evaluating computer-supported cooperative work: Models and frameworks. In *CSCW'04: Proceedings of the 2004 ACM conference on Computer-supported cooperative work*, pages 112–121, Chicago, IL, USA, November 2004. ACM Press. ISBN 1-58113-810-5.

Allen Newell. *Unified theories of cognition*. Harvard University Press, Cambridge, MA, USA, 1994. ISBN 0-674-92101-1.

David Nguyen and John Canny. MultiView: Improving trust in group video conferencing through spatial faithfulness. In *CHI'07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1465–1474, San Jose, CA, USA, April 2007. ACM Press. ISBN 978-1-59593-593-9.

Fred Niederman and John Bryson. Influence of computer-based meeting support on process and outcomes for a divisional coordinating group. *Group Decision and Negotiation*, 7(4):293–325, July 1998. ISSN 0926-2644.

Jakob Nielsen. Finding usability problems through heuristic evaluation. In *CHI'92: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 373–380, Monterey, CA, USA, May 1992. ACM Press. ISBN 0-89791-513-5.

Chris Nodder, Gayna Williams, and Deborah Dubrow. Evaluating the usability of an evolving collaborative product: Changes in user type, tasks and evaluation methods over time. In *GROUP'99: Proceedings of the international ACM SIGGROUP conference on Supporting group work*, pages 150–159, Phoenix, AZ, USA, November 1999. ACM Press. ISBN 1-58113-065-1.

Jay F. Nunamaker, Alan R. Dennis, Joseph S. Valacich, Douglas R. Vogel, and Joey F. George. Electronic meeting systems to support group work. *Communications of the ACM*, 34(7):40–61, July 1991. ISSN 0001-0782.

Jay F. Nunamaker, Robert O. Briggs, Daniel D. Mittleman, Douglas R. Vogel, and Pierre A. Balthazard. Lessons from a dozen years of group support systems research: A discussion of lab and field findings. *Journal of Management Information Systems*, 13(3):163–207, December 1996. ISSN 0742-1222.

Judith Olson and Gary Olson. The growth of cognitive modeling in human-computer interaction since GOMS. *Human-Computer Interaction*, 5(2): 221–265, 1990. ISSN 0737-0024.

Alex F. Osborn. *Applied imagination: Principles and procedures of creative problem-solving*. Scribner, New York, NY, USA, third edition, 1963. ISBN 684-41393-0.

Michael Parent and R. Brent Gallupe. The role of leadership in group support systems failure. *Group Decision and Negotiation*, 10(1):405–422, January 2001. ISSN 0926-2644.

Sharoda A. Paul and Meredith R. Morris. CoSense: Enhancing sensemaking for collaborative web search. In *CHI'09: Proceedings of the twenty-seventh international conference on Human factors in computing systems*, pages

1771–1780, Boston, MA, USA, April 2009. ACM Press. ISBN 978-1-60558-246-7.

Udai S. Pawar, Joyojeet Pal, Rahul Gupta, and Kentaro Toyama. Multiple mice for retention tasks in disadvantaged schools. In *CHI'07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1581–1590, San Jose, CA, USA, April 2007. ACM Press. ISBN 978-1-59593-593-9.

Chengzhi Peng. Survey of collaborative drawing support tools. *Computer Supported Cooperative Work*, 1(3):197–228, September 1993. ISSN 0925-9724.

David Pinelle and Carl Gutwin. A review of groupware evaluations. In *WETICE'00: Proceedings of the ninth IEEE international workshops on Enabling technologies*, pages 86–91, Gaithersburg, MD, USA, June 2000. IEEE Press. ISBN 0-7695-0798-0.

David Pinelle and Carl Gutwin. Groupware walkthrough: Adding context to groupware usability evaluation. In *CHI'02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 455–462, Minneapolis, MN, USA, April 2002. ACM Press. ISBN 1-58113-453-3.

David Pinelle and Carl Gutwin. Evaluating teamwork support in tabletop groupware applications using collaboration usability analysis. *Personal and Ubiquitous Computing*, 12(3):237–254, January 2008. ISSN 1617-4909.

David Pinelle, Carl Gutwin, and Saul Greenberg. Task analysis for groupware usability evaluation: Modeling shared-workspace tasks with the mechanics of collaboration. *ACM Transactions on Computer-Human Interaction*, 10 (4):281–311, December 2003. ISSN 1073-0516.

David Pinelle, Mutasem Barjawi, Miguel Nacenta, and Regan Mandryk. An evaluation of coordination techniques for protecting objects and territories in tabletop groupware. In *CHI'09: Proceedings of the twenty-seventh international conference on Human factors in computing systems*, pages 2129–2138, Boston, MA, USA, April 2009. ACM Press. ISBN 978-1-60558-246-7.

Anne M. Piper and James D. Hollan. Supporting medical conversations between deaf and hearing individuals with tabletop displays. In *CSCW'08: Proceedings of the 2008 ACM conference on Computer-supported cooperative work*, pages 147–156, San Diego, CA, USA, November 2008. ACM Press. ISBN 978-1-60558-007-4.

Anne M. Piper, Eileen O'Brien, Meredith R. Morris, and Terry Winograd. SIDES: A cooperative tabletop computer game for social skills development. In *CSCW'06: Proceedings of the 2006 ACM conference on Computer-supported cooperative work*, pages 1–10, Banff, Canada, November 2006. ACM Press. ISBN 1-59593-249-6.

Lydia Plowman, Yvonne Rogers, and Magnus Ramage. What are workplace studies for? In *ECSCW'95: Proceedings of the fourth european conference on Computer-supported cooperative work*, pages 309–324, Stockholm, Sweden, September 1995. Kluwer. ISBN 0-7923-3697-6.

Peter G. Polson, Clayton Lewis, John Rieman, and Cathleen Wharton. Cognitive walkthroughs: A method for theory-based evaluation of user interfaces. *International Journal of Man-Machine Studies*, 36(5):741–773, 1992. ISSN 0020-7373.

Thorsten Prante, Carsten Magerkurth, and Norbert Streitz. Developing CSCW tool for idea finding: Empirical results and implications for design. In *CSCW'02: Proceedings of the 2002 ACM conference on Computer-supported cooperative work*, pages 106–115, New Orleans, LA, USA, November 2002. ACM Press. ISBN 1-58113-560-2.

Roger S. Pressman. *Software engineering: A practitioner's approach.* McGraw-Hill, New York, NY, USA, 2001. ISBN 0-07-365578-3.

Wolfgang Prinz and Sabine Kolvenbach. Support for workflows in a ministerial environment. In *CSCW'96: Proceedings of the 1996 ACM conference on Computer-supported cooperative work*, pages 199–208, Boston, MA, USA, November 1996. ACM Press. ISBN 0-89791-765-0.

Gitesh K. Raikundalia and Hao L. Zhang. Newly-discovered group awareness mechanisms for supporting real-time collaborative authoring. In *AUIC'05:*

Proceedings of the sixth Australasian conference on User interface, pages 127–136, Newcastle, Australia, January 2005. Australian Computer Society. ISBN 1-920682-22-8.

Sonya Rajan, Scotty D. Craig, Barry Gholson, Natalie K. Person, and Arthur C. Graesser. AutoTutor: Incorporating back-channel feedback and other human-like conversational behaviors into an intelligent tutoring system. *International Journal of Speech Technology*, 4(2):117–126, 2001. ISSN 1381-2416.

Abhishek Ranjan, Jeremy P. Birnholtz, and Ravin Balakrishnan. An exploratory analysis of partner action and camera control in a video-mediated collaborative task. In *CSCW'06: Proceedings of the 2006 ACM conference on Computer-supported cooperative work*, pages 403–412, Banff, Canada, November 2006. ACM Press. ISBN 1-59593-249-6.

Ronald A. Rensink. Change detection. *Annual Review of Psychology*, 53(1): 245–277, 2002. ISSN 0066-4308.

Claudia Roda and Julie Thomas. Attention aware systems: Introduction to special issue. *Computers in Human Behavior*, 22(4):555–556, July 2006. ISSN 0747-5632.

Yvonne Rogers. Exploring obstacles: Integrating CSCW in evolving organisations. In *CSCW'94: Proceedings of the 1994 ACM conference on Computer-supported cooperative work*, pages 67–77, Chapel Hill, NC, USA, October 1994. ACM Press. ISBN 0-89791-689-1.

Mark Roseman and Saul Greenberg. Building real-time groupware with GroupKit, a groupware toolkit. *ACM Transactions on Computer-Human Interaction*, 3(1):66–106, March 1996. ISSN 1073-0516.

Vassil Roussev, Prasun Dewan, and Vibhor Jain. Composable collaboration infrastructures based on programming patterns. In *CSCW'00: Proceedings of the 2000 ACM conference on Computer-supported cooperative work*, pages 117–126, Philadelphia, PA, USA, December 2000. ACM Press. ISBN 1-58113-222-0.

Dennis W. Rowe, John Sibert, and Don Irwin. Heart rate variability: Indicator of user state as an aid to human-computer interaction. In *CHI'98: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 480–487, Los Angeles, CA, USA, April 1998. ACM Press. ISBN 0-201-30987-4.

Mark S. Sanders and Ernest J. McCormick. *Human factors in engineering and design*. McGraw-Hill, New York, NY, USA, seventh edition, 1992. ISBN 0-07-112826-3.

Christopher Saunders. Get your IM in shape: A workplace guide for 2004. *Datamation*, January 2004. URL `http://itmanagement.earthweb.com/entdev/article.php/3294811`. Retrieved May 2009.

Kjeld Schmidt. The problem with awareness. *Computer Supported Cooperative Work*, 11(3-4):285–298, September 2002. ISSN 0925-9724.

Jean Scholtz and Michelle P. Steves. A framework for real-world software systems evaluations. In *CSCW'04: Proceedings of the 2004 ACM conference on Computer-supported cooperative work*, pages 600–603, Chicago, IL, USA, November 2004. ACM Press. ISBN 1-58113-810-5.

Stacey D. Scott, Karen D. Grant, and Regan L. Mandryk. System guidelines for co-located, collaborative work on a tabletop display. In *ECSCW'03: Proceedings of the eighth european conference on Computer-supported cooperative work*, pages 159–178, Helsinki, Finland, September 2003. Springer. ISBN 978-1-4020-1573-1.

Michael Scriven. *Evaluation thesaurus*. Edge Press, Point Reyes Station, CA, USA, third edition, 1981. ISBN 0-918528-18-6.

Hanna M. Söderholm, Diane H. Sonnenwald, Bruce Cairns, James E. Manning, Greg F. Welch, and Henry Fuchs. The potential impact of 3D telepresence technology on task performance in emergency trauma care. In *GROUP'07: Proceedings of the 2007 international ACM conference on Supporting group work*, pages 79–88, Sanibel Island, FL, USA, November 2007. ACM Press. ISBN 978-1-59593-845-9.

Kimron L. Shapiro, Karen M. Arnell, and Jane E. Raymond. The attentional blink. *Trends in Cognitive Sciences*, 1(8):291–296, 1997. ISSN 1364-6613.

Duncan Shaw, Colin Eden, and Fran Ackerman. Evaluating group support systems: Improving brainstorming research methodology. RP 02-09, Aston Business School Research Institute, Birmingham, UK, May 2002.

Herbert A. Simon. Designing organizations for an information-rich world. In Martin Greenberger, editor, *Computers, communication, and the public interest*, pages 37–72. Johns Hopkins University Press, Baltimore, MD, USA, 1971. ISBN 0-8018-1135-X.

Daniel J. Simons and Ronald A. Rensink. Change blindness: Past, present, and future. *Trends in Cognitive Sciences*, 9(1):16–20, 2005. ISSN 1364-6613.

Markus Sohlenkamp and Greg Chwelos. Integrating communication, cooperation, and awareness: The DIVA virtual office environment. In *CSCW'94: Proceedings of the 1994 ACM conference on Computer-supported cooperative work*, pages 331–343, Chapel Hill, NC, USA, October 1994. ACM Press. ISBN 0-89791-689-1.

Cheri Speier, Joseph S. Valacich, and Iris Vessey. The influence of task interruption on individual decision making: An information overload perspective. *Decision Sciences*, 30(2):337–360, 1999. ISSN 0011-7315.

M. Stefik, Daniel G. Bobrow, Stan Lanning, Deborah Tatar, and Greg Foster. WYSIWIS revised: Early experiences with multi-user interfaces. In *CSCW'86: Proceedings of the 1986 ACM conference on Computer-supported cooperative work*, pages 276–290, Austin, TX, USA, December 1986. ACM Press. ISBN 1-23-456789-0.

Robert J. Sternberg. *Cognitive psychology*. Wadsworth, Belmont, CA, USA, third edition, 2003. ISBN 0-155-08535-2.

Oliver Stiemerling and Armin B. Cremers. The use of cooperation scenarios in the design and evaluation of a CSCW system. *IEEE Transactions on Software Engineering*, 24(12):1171–1181, December 1998. ISSN 0098-5589.

Dane Stuckel and Carl Gutwin. The effects of local lag on tightly-coupled interaction in distributed groupware. In *CSCW'08: Proceedings of the 2008 ACM conference on Computer-supported cooperative work*, pages 447–456, San Diego, CA, USA, November 2008. ACM Press. ISBN 978-1-60558-007-4.

Masanori Sugimoto, Kazuhiro Hosoi, and Hiromichi Hashizume. Caretta: A system for supporting face-to-face collaboration by integrating personal and shared spaces. In *CHI'04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 41–48, Vienna, Austria, April 2004. ACM Press. ISBN 1-58113-702-8.

Chengzheng Sun. Undo any operation at any time in group editors. In *CSCW'00: Proceedings of the 2000 ACM conference on Computer-supported cooperative work*, pages 191–200, Philadelphia, PA, USA, December 2000. ACM Press. ISBN 1-58113-222-0.

Margaret H. Szymanski, Paul M. Aoki, Rebecca E. Grinter, Amy Hurst, James D. Thornton, and Allison Woodruff. Sotto Voce: Facilitating social learning in a historic house. *Computer Supported Cooperative Work*, 17 (1):5–34, February 2008. ISSN 0925-9724.

Hamdy A. Taha. *Operations research: An introduction*. Prentice Hall, Upper Saddle River, NJ, USA, seventh edition, 2003. ISBN 0-13-048808-9.

Susan G. Tammaro, Jane N. Mosier, Nancy C. Goodwin, and G. Spitz. Collaborative writing is hard to support: A field study of collaborative writing. *Computer Supported Cooperative Work*, 6(1):19–51, March 1997. ISSN 0925-9724.

Anthony Tang, Melanie Tory, Barry Po, Petra Neumann, and Sheelagh Carpendale. Collaborative coupling over tabletop displays. In *CHI'06: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1181–1190, Montreal, Canada, April 2006. ACM Press. ISBN 1-59593-178-3.

Charlotte Tang and Sheelagh Carpendale. Evaluating the deployment of a mobile technology in a hospital ward. In *CSCW'08: Proceedings of*

the 2008 ACM conference on Computer-supported cooperative work, pages 205–214, San Diego, CA, USA, November 2008. ACM Press. ISBN 978-1-60558-007-4.

Konrad Tollmar, Ovidiu Sandor, and Anna Schömer. Supporting social awareness at work: Design and experience. In *CSCW'96: Proceedings of the 1996 ACM conference on Computer-supported cooperative work*, pages 298–307, Boston, MA, USA, November 1996. ACM Press. ISBN 0-89791-765-0.

Zachary O. Toups, Andruid Kerne, William Hamilton, and Alan Blevins. Emergent team coordination: From fire emergency response practice to a non-mimetic simulation game. In *GROUP'09: Proceedings of the ACM 2009 international conference on Supporting group work*, pages 341–350, Sanibel Island, FL, USA, May 2009. ACM Press. ISBN 978-1-60558-500-0.

Theophanis Tsandilas and Ravin Balakrishnan. An evaluation of techniques for reducing spatial interference in single display groupware. In *ECSCW'05: Proceedings of the ninth european conference on Computer-supported cooperative work*, pages 225–245, Paris, France, September 2005. Springer. ISBN 1-4020-4022-9.

Pihlip Tuddenham and Peter Robinson. Territorial coordination and workspace awareness in remote tabletop collaboration. In *CHI'09: Proceedings of the twenty-seventh international conference on Human factors in computing systems*, pages 2139–2148, Boston, MA, USA, April 2009. ACM Press. ISBN 978-1-60558-246-7.

Lai-lai Tung and Efraim Turban. A proposed research framework for distributed group support systems. *Decision Support Systems*, 23(2):175–188, June 1998. ISSN 0167-9236.

Michael Twidale, David Randall, and Richard Bentley. Situated evaluation for cooperative systems. In *CSCW'94: Proceedings of the 1994 ACM conference on Computer-supported cooperative work*, pages 441–452, Chapel Hill, NC, USA, October 1994. ACM Press. ISBN 0-89791-689-1.

Andrew H. Van de Ven and André L. Delbecq. Determinants of coordination modes within organizations. *Americal Sociological Review*, 41(2):322–338, April 1976. ISSN 0003-1224.

Julita Vassileva and Lingling Sun. Evolving a social visualization design aimed at increasing participation in a class-based online community. *International Journal of Cooperative Information Systems*, 17(4):443–466, December 2008. ISSN 0218-8430.

Alonso H. Vera, Bonnie E. John, Roger Remington, Michael Matessa, and Michael A. Freed. Automating human-performance modeling at the millisecond level. *Human-Computer Interaction*, 20(3):225–265, 2005. ISSN 0737-0024.

Roel Vertegaal. The GAZE groupware system: Mediating joint attention in multiparty communication and collaboration. In *CHI'99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 294–301, Pittsburgh, PA, USA, May 1999. ACM Press. ISBN 0-201-48559-1.

Roel Vertegaal. Attentive user interfaces: Introduction. *Communications of the ACM*, 46(3):31–33, March 2003. ISSN 0001-0782.

Roel Vertegaal, Ivo Weevers, Changuk Sohn, and Chris Cheung. GAZE-2: Conveying eye contact in group video conferencing using eye-controlled camera direction. In *CHI'03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 521–528, Ft. Lauderdale, FL, USA, April 2003. ACM Press. ISBN 1-58113-630-7.

Roel Vertegaal, Jeffrey S. Shell, Daniel Chen, and Aadil Mamuji. Designing for augmented attention: Towards a framework for attentive user interfaces. *Computers in Human Behavior*, 22(4):771–789, July 2006. ISSN 0747-5632.

Yao Wang, Wolfang Gräther, and Wolfgang Prinz. Suitable notification intensity: The dynamic awareness system. In *GROUP'07: Proceedings of the 2007 international ACM conference on Supporting group work*, pages 99–106, Sanibel Island, FL, USA, November 2007. ACM Press. ISBN 978-1-59593-845-9.

Justin D. Weisz. Synchronous broadcast messaging: The use of ICT. In *CHI'06: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1293–1302, Montreal, Canada, April 2006. ACM Press. ISBN 1-59593-178-3.

S.A. Wensveen, J.P. Djajadiningrat, and C.J. Overbeeke. Interaction frogger: A design framework to couple action and function through feedback and feedforward. In *DIS'04: Proceedings of the 2004 conference on Designing interactive systems*, pages 177–184, Cambridge, MA, USA, July 2004. ACM Press. ISBN 1-58113-787-7.

Steve Whittaker and Brian Amento. Seeing what you are hearing: Coordinating responses to trouble reports in network troubleshooting. In *ECSCW'03: Proceedings of the eighth european conference on Computer-supported cooperative work*, pages 219–238, Helsinki, Finland, September 2003. Springer. ISBN 978-1-4020-1573-1.

Christopher D. Wickens and Justin G. Hollands. *Engineering psychology and human performance*. Prentice Hall, Upper Saddle River, NJ, USA, third edition, 2000. ISBN 0-321-04711-7.

Christopher D. Wickens and Jason S. McCarley. *Applied attention theory*. CRC Press, Boca Raton, FL, USA, 2008. ISBN 0-8058-5983-7.

Kin F. Wong. The relationship between attentional blink and psychological refractory period. *Journal of Experimental Psychology: Human Perception and Performance*, 28(1):54–71, February 2002. ISSN 0096-1523.

Bruce Woodcock. An analysis of MMOG subscription growth, May 2008. URL `http://www.2008.loginconference.com/session.php?id=53692`. Lecture presented at the 2008 ION Game conference. Retrieved November 2008.

Huahai Yang and Gary M. Olson. Exploring collaborative navigation: The effect of perspectives on group performance. In *CVE'02: Proceedings of the fourth international conference on Collaborative virtual environments*, pages 135–142, Bonn, Germany, September 2002. ACM Press. ISBN 1-58113-489-4.

Nicole Yankelovich, William Walker, Patricia Roberts, Mike Wessler, Jonathan Kaplan, and Joe Provino. Meeting central: Making distributed meetings more effective. In *CSCW'04: Proceedings of the 2004 ACM conference on Computer-supported cooperative work*, pages 419–428, Chicago, IL, USA, November 2004. ACM Press. ISBN 1-58113-810-5.

Yufei Yuan, Milena Head, and Mei Du. The effects of multimedia communication on web-based negotiation. *Group Decision and Negotiation*, 12(2): 89–109, March 2003. ISSN 0926-2644.

Ana Zanella and Saul Greenberg. Reducing interference in single display groupware through transparency. In *ECSCW'01: Proceedings of the sixth european conference on Computer-supported cooperative work*, pages 339–358, Bonn, Germany, September 2001. Springer. ISBN 978-0-7923-7163-2.

Shumin Zhai. What's in the eyes for attentive input. *Communications of the ACM*, 46(3):34–39, March 2003. ISSN 0001-0782.

Qiang A. Zhao and John T. Stasko. Evaluating image filtering based techniques in media space applications. In *CSCW'98: Proceedings of the 1998 ACM conference on Computer-supported cooperative work*, pages 11–18, Seattle, WA, USA, November 1998. ACM Press. ISBN 1-58113-009-0.

Acronyms

AB	Attentional Blink (p. 119)
AUI	Attentive User Interface (p. 121)
CATHCI	Cognitive Analysis Tool for HCI (p. 39)
CB	Change Blindness (p. 120)
CMN-GOMS	Card, Moran, Newell GOMS (p. 38)
CPM-GOMS	Cognitive, Perceptual, Motor GOMS (p. 42)
CRITIQUE	Convenient, Rapid, Interactive Tool for Integrating Quick Usability Evaluations (p. 37)
DGOMS	Distributed GOMS (p. 45)
DTB	Defer-To-Boundary (p. 129)
GLEAN	GOMS Language and Evaluation ANalysis (p. 41)
GOMS	Goals, Operators, Methods, and Selection rules (p. 35)
GU	Goal Unit (p. 95)
HCI	Human-Computer Interaction
ISI	InterStimulus Interval (p. 120)
KLM	Keystroke-Level Model (p. 36)
LTM	Long-Term Memory (p. 35)
MAUI	Multi-User Awareness UI Toolkit (p. 114)
MHP	Model Human Processor (p. 34)
NGOMSL	Natural GOMS Language (p. 39)
PERT	Program Evaluation and Review Technique (p. 42)

PRP Psychological Refractory Period (p. 119)

QGOMS Quick GOMS (p. 39)

SDG Single Display Groupware (p. 94)

SOA Stimulus Onset Asynchrony (p. 120)

SQFD Software Quality Function Deployment (p. 85)

TCP/IP Transport Control Protocol/Internet Protocol (p. 139)

UI User Interface

VISNU Validation of Intelligent Systems aNd Usability (p. 39)

WM Working Memory (p. 34)

XML eXtensible Markup Language (p. 139)