

Role-Based Control of Shared Application Views

by

Lior Berry

B.Sc., Tel Aviv University, 1998

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

THE FACULTY OF GRADUATE STUDIES

(Computer Science)

THE UNIVERSITY OF BRITISH COLUMBIA

August 19, 2005

© Lior Berry, 2005

Abstract

Collaboration often relies on all group members having a shared view of a single-user application. A common situation is a single active presenter sharing a live view of her workstation screen with a passive audience, using simple hardware-based video signal projection onto a large screen or simple bitmap-based sharing protocols. This offers simplicity and some advantages over more sophisticated software-based replication solutions, but everyone has the exact same view of the application. This conflicts with the presenter's need to keep some information and interaction details private. It also fails to recognize the needs of the passive audience, who may struggle to follow the presentation because of the amount of interaction details, display clutter or insufficient familiarity with the application.

Views that cater to the different roles of the presenter and the audience can be provided by custom solutions, but these tend to be bound to a particular application. This thesis describes a general technique and implementation details of a prototype system that allows standardized role-specific views of existing single-user applications and permits additional customization that is application-specific with no change to the application source code. Role-based policies control manipulation and display of shared windows and image buffers produced by the application, providing semi-automated privacy protection, relaxed verbosity and added visual cues to meet both presenter and audience needs.

The system's prototype was evaluated in a formal user study, in a task training scenario using a shared view. The study showed that adding visual cues improves accuracy, while privacy filters do not result in performance penalties but can even assist viewers.

Contents

Abstract	ii
Contents	iii
List of Tables	vii
List of Figures	viii
Acknowledgements	x
1 Introduction	1
1.1 Sharing an application View	2
1.2 Motivation	4
1.3 Our contribution	7
1.4 Thesis Outline	8
2 Privacy and Augmentation Problems	9
2.1 Scenario: Semi-structured presentation	10
2.2 Scenario: Brain-storming with multiple content sources	11
2.3 Scenario: Access Control for desktop sharing	12
2.4 Privacy Concerns	13
2.4.1 Privacy vs. Security	14
2.4.2 Privacy properties of presentation scenarios	14
2.4.3 Privacy Risk Management	16
2.4.4 Private Information Sources in View Sharing	17
2.5 Improving the Audience Experience	19
2.5.1 Inadequacy of single-user GUIs for passive viewers	20
2.5.2 Controlling verbosity	22

2.5.3	Mitigating visual clutter	23
3	Related Work and Literature	25
3.1	Collaboration-Aware Solutions	25
3.2	Collaboration-Transparent Solutions	27
3.2.1	Centralized tools	28
3.2.2	Replication-based tools	29
3.3	Screen Recording tools	31
3.4	Spatial and Window Set Manipulations	32
3.5	Visual manipulations	33
3.6	Multi-Machine User Interfaces solutions	35
3.7	Single Display Privacyware	35
3.8	Presentation tools	36
3.9	Presentation Authoring	36
3.10	Animation	38
4	System Description	41
4.1	Core Functionality and Components	41
4.1.1	Cloning Windows	41
4.1.2	“Semantic glue” queries	43
4.1.3	Plug-In Architecture	45
4.1.4	Policies and rules	52
4.2	Manipulating the Visual Representation	55
4.2.1	Blurring	55
4.2.2	Salience and highlighting	57
4.2.3	Spatial manipulations	58
4.2.4	Temporal manipulations	59
4.2.5	Handling Menus	60
4.2.6	Mouse Cursor manipulations	61
4.3	Access Control Extension for Input	61
4.4	Feedback and Control	62
4.4.1	Radar View	62
4.4.2	Changing Privacy Classification	63

4.4.3	Plug-In UI	63
4.4.4	Audience input	64
4.5	Limitations	66
4.5.1	Identifying private information	66
4.5.2	Working on the image buffer	67
4.5.3	Performance	67
4.5.4	Feedback for the presenter	68
5	User Study	75
5.1	Methodology	76
5.1.1	Experimental Design	78
5.2	Method	79
5.2.1	Participants	79
5.2.2	Instruments and Data Collection	79
5.2.3	Procedure	82
5.3	Results	83
5.3.1	Measuring Performance	83
5.3.2	Quantitative Analysis	84
5.3.3	Questionnaire Analysis	92
5.4	Summary and conclusions	98
5.4.1	Effects on performance	99
5.4.2	Balancing privacy and augmentations	100
5.4.3	Perceived utility	101
6	Future Work and Conclusions	102
6.1	Future Work	102
6.1.1	System improvements	102
6.1.2	Future studies	104
6.2	Conclusions	105
	Bibliography	109
A	Questionnaire	114

B Task Descriptions	117
B.1 Marking Scheme	117
B.2 Task descriptions	120

List of Tables

1.1	Categories of View Sharing	2
4.1	The Plug-In API	49
5.1	Subject familiarity with Excel functionality	80
5.2	Performance - Repeated Measures ANOVA	85
5.3	Overall functionality training effectiveness	95

List of Figures

1.1	The unexpected “hazards” of sharing a desktop	5
2.1	Privacy Concerns in a Collaborative Session	10
2.2	Sharing content from multiple laptops	12
2.3	Fine Grained Access Control	13
2.4	Perceptual gap between presenter and viewers	21
2.5	Inadequacy of conventional GUI for Passive Viewers	22
3.1	Custom GroupKit controls to assist viewers	26
3.2	Design Space of Collaborative Applications	27
3.3	Manual control of shared views	39
3.4	Sharing screens for awareness	40
3.5	Visual Surface Manipulations	40
4.1	System Architecture	42
4.2	Use of Extended Desktop Mode	43
4.3	Using Accessibility Information	51
4.4	Obtaining menu information	52
4.5	Semantic driven blurring (Spreadsheet)	55
4.6	Miscellaneous Blurring Examples	56
4.7	Spatial Manipulations	59
4.8	Temporal Manipulations – Iconic Indicators	69
4.9	Manipulating Menus	70
4.10	Overriding Cursors	71
4.11	Radar View	72
4.12	PrivacyControls	73
4.13	Highlighting Examples	74

5.1	Subject Excel Expertise	80
5.2	Task Accuracy	86
5.3	Effects of condition on speed	87
5.4	Effects of condition on efficiency	88
5.5	$P \times A$ interaction	89
5.6	$TASK \times P \times A$ interaction	90
5.7	Condition effects on accuracy per task	91
5.8	Training Experience I	94
5.9	Training Effectiveness	95
5.10	Feature ratings	97

Acknowledgements

I would like to thank my supervisory committee Dr. Kellog Booth, Dr. Lyn Bartram and Dr. Brian Fisher for providing useful feedback and advice and keeping me intrigued all along the way. I would also like to thank Gail Murphy (the second reader) and Ritchie Argue for providing helpful comments on the last versions of the thesis, Maureen Stone for her resourceful comments early on and Barry Po for his tips and assistance. I also thank the members of the Imager Lab and the Interaction Design Reading Group (IDRG). Funding for this research was provided by the Natural Sciences and Engineering Research Council of Canada under its Discovery Grant and Research Network Grant programs, through NECTAR, the Network for Effective Collaboration Technology through Advanced Research. Facilities and research infrastructure were provided by the Canadian Foundation for Innovation awards, including the WestGrid high performance computing initiative.

Dedicated to my beloved wife, Tamar, and my children, Adi and Tal, who supported me in this journey.

In memory of my parents, Tamar and Moshe.

LIOR BERRY

*The University Of British Columbia
August 2005*

Chapter 1

Introduction

People working in groups increasingly rely on the ability to share views of an entire work session or a specific application for co-located or distributed cooperative work. Although new tools and frameworks introduced in recent years support a wide range of collaboration formats, the dominant format is still that of a single person, the *presenter*, sharing a view of her workstation while others, the *audience*, watch. This *generalized presentation* setting applies to people giving conference or classroom presentations, demonstrating software, training others, or engaging in collaborative work where a shared document on a public display is the focus for group discussion. It differs from traditional presentations in its *ad hoc*, extemporaneous nature, and that it is not restricted to applications that are specifically designed for presentations, but instead utilizes any and all of the applications used in normal day-to-day work. Unlike traditional presentations, where the presenter prepares all of the material in advance and may even have one or more rehearsals with “safe” audiences, the generalized presentation is akin to “performing without a net”. The goal of the research described in this thesis is to support this style of shared viewing during collaboration.

Other more symmetrical patterns of collaboration are also possible, where all participants can equally interact with a shared application. However, this mode is far less common. There are technological hurdles to adapting existing desktop tools for more elaborate collaborative modes. There are also social considerations that come into play: people do not easily move away from tools with which they are already familiar. The generalized presentation scenario mimics the teacher-class model and relies on an intrinsic asymmetry of roles and needs: viewers in the audience expect to be passive; the presenter is active. Our focus is on supporting the asymmetric roles in a generalized presentation by adapting the full spectrum of single-user applications to shared viewing for collaborative use.

The assumption of asymmetric roles may seem unrealistic, but even in scenarios such as collaborative editing, where roles are more equal, there is usually a turn-taking pattern.

		Time	
Place		Same	Different
	Same	Face-to-face meeting - shared display (large room display or workstation screen)	Sharing a copy of a document or Video recording of interactions (locally)
	Different	Tele video / desktop conferencing or collaborative synchronous editors	Sharing a copy of a document or Video recording of interactions (over the network)

Table 1.1: View sharing options categorized based on time and place (adapted from Grudin’s groupware options model, 1994).

So at any point in time there is a single editor (the presenter) and all others are just viewers (the audience).

There is a range of solutions to facilitate a shared view of an application or work session. An important distinction is whether a solution is based on the *collaboration-transparency* principle and works with *collaboration-unaware applications*. These solutions do not require the application to know a view of it is being shared and hence no source code changes to the applications are necessary. This stands in contrast to collaboration-aware applications that are specifically designed to support cooperative work and shared views.

Another important distinction is the synchronous or asynchronous nature of the solution (*same time* or *different time*) and whether the solution works for *co-located* or *distributed* participants (see Table 1.1).

We chose to focus our work on collaboration-unaware solutions, mostly because people tend to share existing end-user tools that have no built-in collaboration or view sharing capabilities. We also chose to focus on synchronous, co-located solutions that most closely correspond to the general presentation scenarios. We will argue in Chapter 6, however, that our approach can be extended to synchronous distributed solutions and to asynchronous use of screen recordings.

1.1 Sharing an application View

For co-located groups, view sharing is usually achieved by replicating the video signal from the presenter’s computer onto an external public display (using a projector or a large screen

display). For distributed groups, bitmap-based screen sharing protocols such as the remote frame buffer protocol used in VNC (Richardson, Stafford-Fraser, Wood, and Hopper 1998) provide view sharing. We will consider both of these to be equivalent, and will refer to them as “bitmap-based screen sharing” without distinguishing between hardware and software implementations, or between analog and digital formats. Functionally, both solutions are equivalent. Essentially, a bitmap that contains a copy of the presenter’s visual display is conveyed via a video cable or over a network onto another display and is kept updated as the presenter makes changes.

These solutions afford important properties that more sophisticated collaboration-aware solutions (discussed in Chapter 3) may be hard-pressed to achieve. We believe that the following properties will continue to make bitmap-based view sharing schemes a favorable and viable solution for the foreseeable future:

- **Collaboration transparency** - collaboration is transparent because any application can be shared without requiring it to know it is shared or requiring code changes
- **Minimal software & hardware demands** - viewers do not need to install the shared application, which is often an unacceptable imposition due to licensing restrictions, computation power limitations, operating system incompatibilities, or similar issues. The only extra hardware required is a projector or a network connection.
- **Synchronized views** - there is no need to synchronize views between the presenter and audience because only one copy of the application is running so everyone is guaranteed to see the same view.

The last point allows *referential transparency*: the presenter and the viewers can easily refer to the same objects in the same location within an application window. Sharing a bitmap of the application view also facilitates easy manual drawing, highlighting, and annotating on top of the view. For these and other reasons, synchronized views are desirable. Our system maintains them at all times, except when an explicit choice has been made to alter the views seen by different participants based on their respective roles in the collaboration.

We allow exceptions to synchronizing views because forcing synchronized views is also the chief drawback of many existing schemes. A strict “What You See Is What I See”

(WYSIWIS) mode in which all viewers are forced to share the exact same view that the presenter sees, despite the roles that they play within the group, conflicts with the different needs of the presenter and audience. A “relaxed WYSIWIS” mode has been suggested in the literature (Stefik, Bobrow, Foster, Lanning, and Tatar 1987) to address this problem. Unfortunately, none of the current bitmap-based solutions implement this successfully (Begole, Rosson, and Shaffer 1999).

In this thesis we present a relaxed WYSIWIS mode that retains all of the advantages of traditional bitmap-base sharing. It addresses two of the main needs: maintaining the presenter’s privacy and adapting views to improve the viewers’ ability to follow the presenter’s interactions.

1.2 Motivation

Before describing our solution, we explain the two problems that led us to develop our techniques. These both occur frequently in generalized presentations. Having introduced the problems we set out to solve, we conclude the chapter with a summary of our contributions and outline the rest of this thesis.

Presenter’s Privacy

The initial problem motivating our work was the lack of support for presenter privacy. While generally interested in sharing a view with her audience, there are often interactions or display components a presenter would like to keep private. These may be interactions with other running applications on the desktop or with parts of the shared application that are deemed private.

Some components may contain embarrassing information (such as a navigation history list or an open Instant Messenger client with incoming messages that may not be appropriate for the audience to see). The exposure of some components may be beyond simple embarrassment. For example, exposing a file open dialog that shows files with sensitive client names in a business meeting or exposing parts of a worksheet with confidential parameters could be fatal to a business relationship. Figure 1.1 provides a graphic example.

The need to limit publicly shared information will only intensify as collaboration tech-

¹See one example at <http://www.flickr.com/photos/digitalweb/3797979/> (last checked August 8th, 2005).

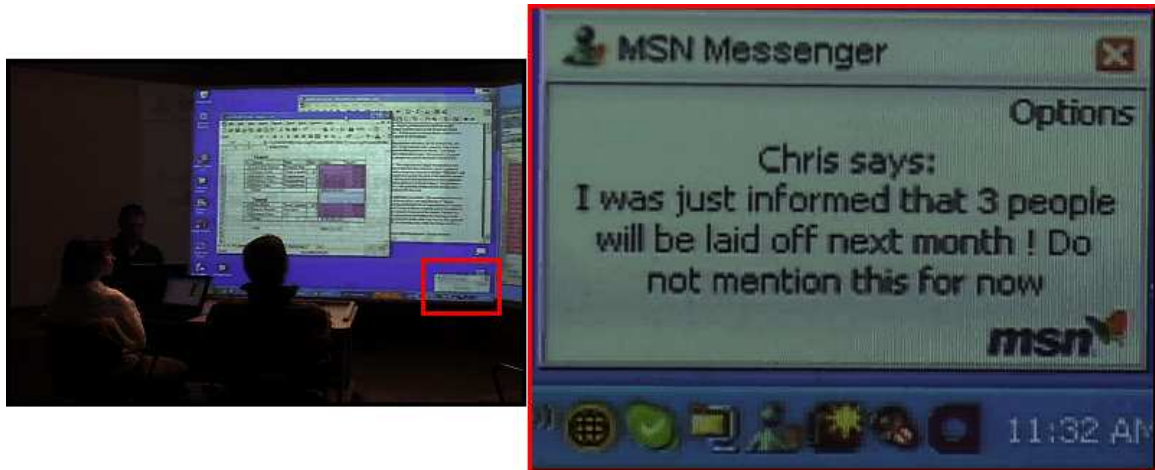


Figure 1.1: Exposing a full desktop is often undesirable. This presenter has left his instant messenger client open when presenting a document from his laptop. A colleague sends him an instant message with sensitive information that is now viewed by all meeting participants. Similar scenarios are quite common in the real world (some are even documented online¹).

nologies and application sharing become part of day-to-day work (both co-located and distributed). It is becoming very common for meeting participants to bring in their laptops and be asked to extemporaneously present materials on a public screen. Presenters may find themselves in an *ad hoc* presentation-like mode without having the time to prepare or while having to do other tasks on their computer in parallel.

Recent work (Hawkey and Inkpen 2005) shows that in similar *ad hoc* scenarios most people would like to take measures to minimize the exposure of private information. (Hawkey focused on accidental exposures of cached web browsing information. It is reasonable to assume the same patterns apply to more sensitive data.)

There are lots of existing mechanisms that can assist users in protecting private information. However, it is not clear that someone engaged in a live presentation can attend effectively to her privacy at the same time she is attending to the quality of the presentation. To make the situation worse, there is also a growing number of applications and components that invade one's desktop autonomously and carry sensitive information: instant messenger arrivals, network notifications, assistive wizards, and a slew of other mechanisms all designed to increase the effectiveness of a single user in dedicated, non-shared activities. These applications may be crucial for the presenter and should be kept active, but must not be exposed to viewers in the audience. Some automation of privacy protection is therefore

called for to enable the presenter to focus on her primary task (the presentation) without worrying about the secondary tasks that are supporting the presentation task.

Assisting Passive Viewers

A different problem arises when we consider the audience experience. Passive viewers may wish to control the type and level of information presented to them and they may require assistive cues to accurately follow the presenter's interactions with an application.

Passive viewers often find themselves visually searching for the current point of interaction. This problem is intensified on large screen displays, where mouse cursors and other UI components are hard to track (Baudisch, Cutrell, and Robertson 2003). In other cases the audience is forced to view tedious interactions by the presenter (such as searching for a menu item or adjusting display parameters) that are irrelevant to their interests and create visual clutter and distraction. Conversely, some manipulations a presenter makes have no associated visual feedback that passive viewers can follow (such as when a "hot key" is used to invoke a command instead of using a menu selection that will be seen on the screen). Viewers will therefore miss important cues in all these situations.

These needs require adapting the shared view to display less information in some cases, and to display more information in other cases. Indeed, existing screen recording tools that were designed to produce training videos, such as Camtasia,² allow visual enhancements and cleanup to be applied to a recorded movie in a separate "post production" editing session. Being able to apply these enhancements in real time and in an automated way would be beneficial for effective collaboration.

Role-Appropriate Views

The two problems presented here, protecting the presenter's privacy and tailoring the audience's experience, are actually both manifestations of the need to provide role-appropriate views of an application to every group member. One approach is to use more elaborate collaboration-aware tools or to adapt existing tools to run in a synchronized mode (Xia et al. 2004), so that presenter and audience have similar but non-identical views.

While custom solutions enable one to flexibly craft views as desired, they fall far short of meeting the advantages of existing bitmap-based protocols that were outlined before

²<http://www.techsmith.com> (last checked August 8th, 2005)

because they usually require all parties to have a copy of the application and they depend strongly on application-specific features. Evidence for this claim is the fact that time and time again people resort to bitmap-based sharing rather than adopting more sophisticated solutions. For this reason we believe that any improvements to bitmap-based sharing modes are highly relevant to improving collaboration “in the field”.

1.3 Our contribution

We have developed a novel framework for adapting a live shared view of applications to meet the presenter’s privacy requirements and to provide viewers with suitable cues and level of detail, balancing concerns for privacy and awareness. Our system uses bitmap-based techniques to transparently share visual information, while allowing policies to be specified that control the generation of different views for the different roles within a collaborating group by reusing the visuals from the running application in a systematic way that does not depend strongly on the inner workings of the application.

To achieve this, the system conducts an “over the shoulder” monitoring of what the presenter is doing, actively manipulating the published visuals in three ways:

- hierarchically based *spatial* transformations for selective sharing, repositioning and scaling of application components (including sub-window regions), or replacement of entire components with standardized iconic representations are applied to the bitmap as it is replicated (or *cloned*) on the shared display;
- simple local or global *chromatic* image filters, such as blurring or highlighting, are applied to the visual surface of the application so that specific components that are most salient to the audience “pop out”, but components that are not become less obvious;
- application-state-based *temporal* transformations are applied to the timeline of captured interactions, such as slowing down interactions or omitting interactions entirely.

To make these manipulations useful and meaningful, some reliance on application-specific semantics is required to extract locations of semantic UI objects and to acquire information about the application’s state. Any such reliance could very well be a “slippery slope” that leads to precisely the application-dependent solutions we seek to avoid. For this

reason we have chosen to limit application dependencies by severely restricting knowledge of applications to certain stylized patterns. This is a critical aspect of our design. It is realized through a plug-in architecture and a set of heuristics for obtaining application semantics without giving up too much generalizability or collaboration transparency (we refer to this part of the system as “*semantic-glue*” and will describe it in detail in Section 4.1.2) .

We have implemented a prototype of the system and have demonstrated how it can be applied to several popular commercial off-the-shelf applications, disproving to some extent the misconception that bitmap-based application sharing forces strict WYSIWIS shared viewing (see the discussion by Begole et al. (1999)). We also conducted a user study for our system that evaluated the balance of privacy and awareness in a training scenario. The results of the study will be used to inform the next iteration in our design. We believe that these results and some of our other conclusions apply equally well to collaboration-aware solutions for shared viewing, even though our focus has been on collaboration-transparent solutions.

1.4 Thesis Outline

In the remaining chapters we describe our work, beginning with the problems of presenter privacy and passive viewer needs for augmentation that were the motivation for our work. These are examined in more detail in Chapter 2 . In Chapter 3 we survey related work and derive guidelines for our system. Chapter 4 describes the prototype system, and some of its advantages and limitations. The user study to evaluate certain aspects of the system is described in Chapter 5. Chapter 6 concludes with possible directions for future work and a discussion of our results and lessons learned from our experience to date. The appendices contain samples of the materials used in the user study and a pointer to an electronic compendium of other materials that are available on the Web from the author.

Chapter 2

Privacy and Augmentation Problems

“sub rosa” — The Romans hung a rose over meetings to indicate the meeting was confidential. Attendees understood that whatever was said under the rose - or sub rosa - had to remain a secret.

— Robert Langdon, “The Da Vinci Code” (Dan Brown).

My fellow journalists called themselves correspondents; I preferred the title of reporter. I wrote what I saw. I took no action - even an opinion is a kind of action.

— Thomas Fowler, “The Quiet American” (Graham Greene).

It is important to first understand the privacy needs of a presenter and the visual augmentation needs of passive viewers in generalized presentations. This chapter discusses both of these after presenting three example scenarios that illustrate some of the motivations for our work.

Throughout this chapter and most of the rest of the thesis we assume that a group of people is collaborating using one or more computers each of which is communicating with the others through a high-speed network. Our main interest is the case of a single computer, which we assume is being used in an “extended desktop” mode (see Figure 4.2) where an auxiliary video output to a shared screen (either a monitor or a projector) displays the contents of the extension to the primary desktop displayed on the main monitor (or LCD screen if the computer is a laptop). We use the term “system” to refer to the prototype implementation of the architecture that we have designed as our solution to the problems we identified in the scenarios that follow.

The scenarios form the basis for the informal set of requirements that was used to develop our prototype.

2.1 Scenario: Semi-structured presentation

Bob and Carol are both managers and Ted and Alice are team members in a group of employees. Bob, the presenter, is discussing the team's budget using a spreadsheet on his laptop that is being projected onto a large shared screen also viewable by Ted and Alice. Carol views Bob's laptop remotely, using VNC. Some of the data, parameters and interactions in Bob's spreadsheet are confidential and should not be exposed to Ted and Alice, but should be available to Bob and Carol (see Figure 2.1). Bob determines some parameters for his spreadsheet using other applications (e.g. an IM client with Carol). A key requirement is that Bob must see the information about the private parameters and he must be able to change them, but without exposing any of the information to Ted or Alice.

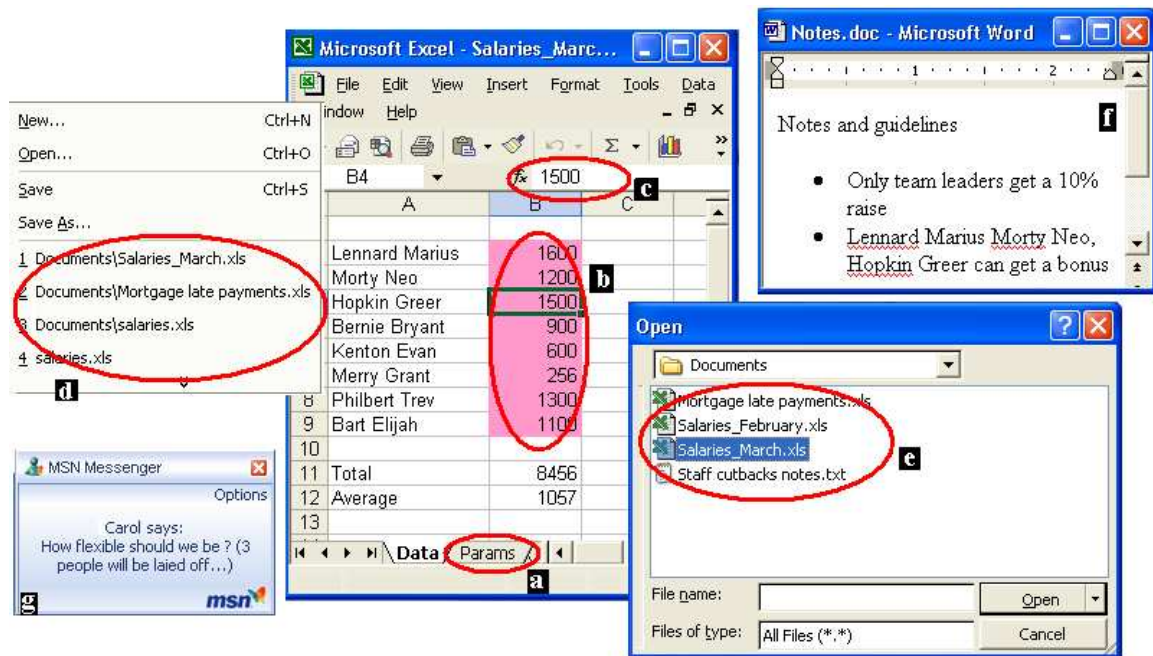


Figure 2.1: Bob's privacy needs in the first scenario: the "Params" worksheet (a) should be entirely private; the entire salaries cell range (b) should be private, as should the copy of the value for Hopkin Greer from within the cell range that is currently displayed in the formula edit box (c); the file dialog (d) and menu (e) expose private file names, which should not be displayed; and Bob's notes (f) and his IM client with Carol (g) should be kept private.

Bob could extract just the relevant data to a new spreadsheet and project it on a shared auxiliary screen with the master spreadsheet visible solely on his laptop. In theory this solves the problem, but not in practice. Extracting the appropriate information with its dependencies is possible (though not trivial). However, synchronizing the spreadsheet

versions when changes are made is time-consuming, error-prone and requires redundant computations (even when using automated scripts). Moreover, Bob will still need to make sure that updates do not reveal private information. The cognitive overhead of managing the session will diminish his ability to focus on the budget. We want a solution that lets Bob concentrate on his primary task (the budget) and not worry about maintaining his privacy.

Carol has different needs. She has to watch Bob's verbose UI interactions, some of which are distracting or take up valuable screen space on the laptop she is using to follow Bob's presentation on his larger monitor screen. Ted and Alice, on the other hand, only see the results of Bob's manipulations echoed in the secondary spreadsheet on the projection screen, so they are missing critical interaction cues that could be crucial to their understanding of the presentation.

This scenario demonstrates the need for role-based viewing policies. Bob and Carol need to see a different view than Ted and Alice, because of privacy concerns. All passive viewers need an augmented, "cleaned up" version of what Bob sees in order to comfortably follow his actions. In fact, more than one is needed because Carol is an expert user with different authorization to see salary information, while Ted and Alice are novices who need help following the spreadsheet manipulations but who have no authorization to see the salary information except in summary form.

2.2 Scenario: Brain-storming with multiple content sources

Huey, Dewey & Louie are conducting a project status report meeting. They are trying to revisit what each has done and lay out future plans for the project. Each of them brings his own laptop and there is a public screen in the room they can all share by connecting their laptops to it through the network. As part of the meeting each participant shares live content from his personal computer with the other people, but constantly interacts with other "private" components on his machine as well. Unfortunately, the public screen is limited in size and cannot show all three of the desktops images at the same time. Therefore, each participant would like to share a view of only a part of an application or a specific document item so that all of the shared information can be seen on the public screen but none of the private information.



Figure 2.2: Each meeting participant shares only part of his desktop on the public screen, using the WinCuts system (picture taken from Tan et al. (2004)). However this system is based on sharing a manually selected region of a window, which quickly breaks down as users resize, scroll or change their selection and focus.

This scenario restates the need for selective sharing of applications. It also demonstrates that even when sharing only a single application, it still has dialogs, menus and wizards or UI elements that take up screen space and clutter the display (intensified in this case when several participants interact at the same time). In many cases sharing only window parts or providing alternative awareness cues can be better.

2.3 Scenario: Access Control for desktop sharing

Adam is a new user of a computer system and has encountered some difficulties filling in fields of a certain dialog box. Adam would like his colleague Eve, who sits in a different office, to fill these parts in.

Adam starts desktop sharing with Eve so she can fill in these fields for him. However Adam would like to make sure that Eve cannot touch or view other dialog fields, that she cannot hit the OK button that releases the dialog before he has had a chance to check what she has done, and that she cannot interact with other applications on his desktop that are not pertinent to the task.

(Alternatively, Adam and Eve might be co-located, so Adam projects his laptop with the application onto a shared display and Eve uses her PDA to control Adam's machine

using a multi-screen collaboration tool (Booth, Fisher, Lin, and Argue 2002). Adam still has the same concerns about what Eve can see and do on his computer.)

This scenario demonstrates the need for fine grained access control in desktop sharing and ubiquitous computing scenarios. In existing solutions whoever gets control over the shared desktop can act as if he is the user currently logged in. We want to do better than this “all or nothing” approach.

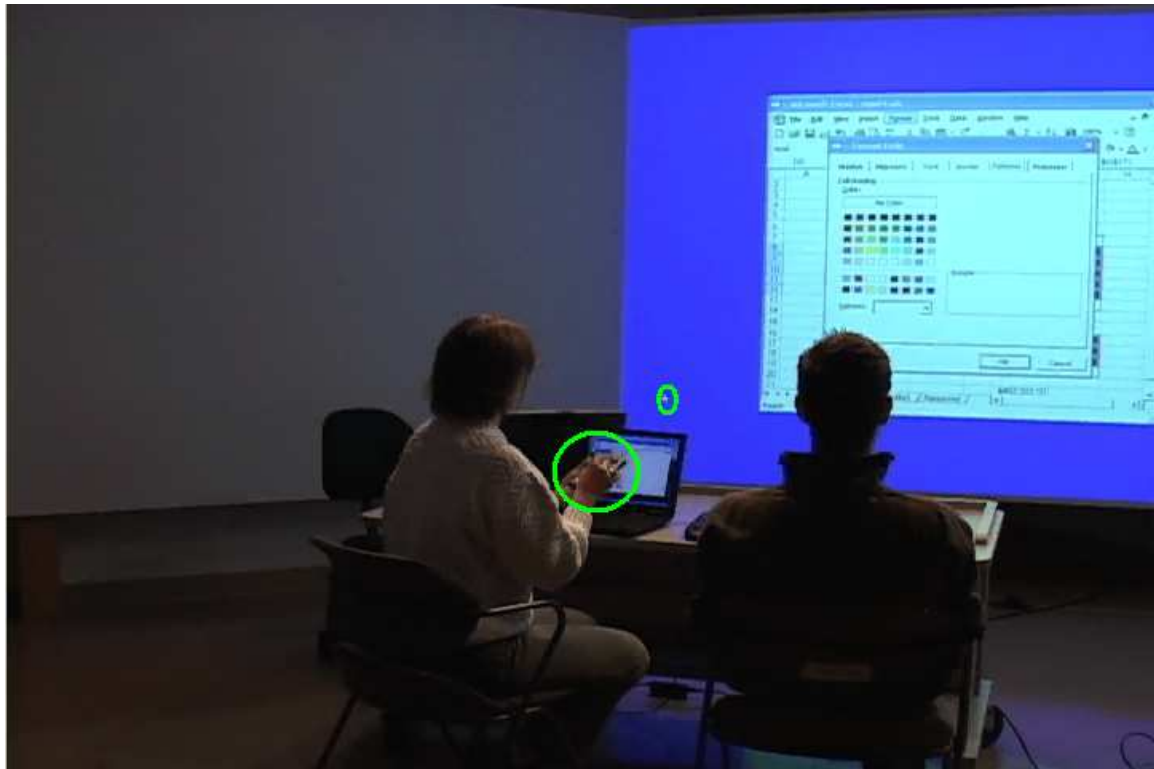


Figure 2.3: Eve is controlling Adam’s machine through her PDA (taken from the video appendix of Berry et al. (2005)). Adam would like to make sure that Eve can only interact with specific dialog elements and cannot hit the OK button.

2.4 Privacy Concerns

Privacy can be an important factor in the adoption of Computer Supported Collaborative tools. Yet, it is hard to come up with a crisp definition of what is considered private information that needs protection, and what information does not. One of the reasons is that privacy is often a highly subjective matter: we each perceive privacy differently according to our values, interests, and power. Another reason is the wide range of conflated issues that

are classified under privacy. These range from individuals withholding activities from other individuals (similar to the presentation scenarios laid out before) to groups keeping secrets from the state, encryption, identity theft, and “big brother” scenarios (Lederer, Mankoff, and Dey 2003).

2.4.1 Privacy vs. Security

It is important to first distinguish between privacy and security. Often these tend to be mixed up. Hong, Ng, Lederer, and Landay (2004) point out that security relates to “mechanisms and techniques that control who may use or modify the computer or the information stored in it,” and privacy relates to “the ability of an individual (or organization) to decide whether, when, and to whom personal (or organizational) information is released”. Obviously the release of some types of private information has security implications, and conversely the application of security measures can affect privacy.

In the context of generalized presentation scenarios, there are many privacy concerns that do not pose a real security threat, but may still put one of the participants (usually the presenter) in an awkward position. As Palen and Dourish (2003) note “... in mundane and pervasive activity like video conferencing, shared calendar management and instant messaging communications, concerns most salient to users include minimizing embarrassment, protecting turf (territoriality) and staying in control of one’s time.” There are also security-related privacy concerns (e.g. accidentally revealing one’s credit card number or user name). Our system attempts to support both types of privacy concerns.

2.4.2 Privacy properties of presentation scenarios

Most privacy related phenomena can be classified as having surveillance related properties or transaction related properties (Lederer et al. 2003), classified by the level of participation and awareness on the subject’s part (the presenter in our case), and by the ability to apply machine-processing to the captured information (also referred to as monitored vs. searched)

Surveillance - is often interpreted as a disempowerment of the subject, who is unaware of the fact that his actions are being recorded by institutionally managed cameras, personal cameras, or overseen or overheard behavior. Closest to our scenarios are video media spaces described by Boyle and Greenberg (2005), in which people can choose whether to keep a

video channel open with colleagues (much like a presenter chooses to share a view of his desktop) for awareness and informal communication.

In these scenarios people still like to keep several privacy properties: *autonomy* (control over identity and self-presentation), *confidentiality* (control over information access and fidelity) and *solitude* (control over interactions and attention). Most of the tension arises when people forget they may be viewed by others and disclose sensitive information or expose themselves in an awkward state (when undressed in an at-home setting or simply when making mistakes that make them look bad in the media space).

Transactions - Transactions are usually associated with data that the subject has willingly agreed to release: for instance credit card transactions, RFID tags, and information entered into HTML forms. The subject has the ability to alter the disclosure by changing the content or conditions of the transaction. The transaction is usually recorded and is subject to a machine-search.

In our shared view scenarios, the presenter willingly chose to expose information visible on his computer to others, but as with transactional types of information he is fully aware of what is shared and has the ability to change it (for instance by deciding what to do with the shared application, by preparing ahead of time or by dragging a window off-screen).

Furthermore, it is very common for shared application view sessions to be recorded. These recordings are more subject to machine search than surveillance videos. Analyzing and searching live video is a hard problem, whereas indexing a screen recording is much more tractable (our system as described in Chapter 4 demonstrates some of these capabilities). Shared application view sessions can be made transactional (Li, Spiteri, Bates, and Hopper 2000), but this does not resolve all of the problems.

Putting the previous observations together we can state that in generalized presentation scenarios a presenter is knowingly sharing with others content that could be sensitive or interactions that can make him look bad, but has at least the theoretical ability to change or limit the exposed information. However, micro-managing all possible privacy leaks while attempting to give a presentation at the same time is not an easy task, as we shall show in the following sections. We therefore believe that automated help is required to meet all of our expectations for privacy during generalized presentations.

2.4.3 Privacy Risk Management

An important aspect when considering the development of a system aimed at privacy protection is to assess the magnitude of the privacy risk and the cost of the solution. Hong et al. (2004) suggest looking at:

- The *likelihood* L that an unwanted disclosure of personal information occurs
- The *damage* D that will happen on such a disclosure
- The *cost* C of adequate privacy protection

Hong concludes that implementing and using a particular privacy solution is worthwhile only if the potential damage outweighs the cost of creating and using the solution ($C < LD$). In many situations $C > LD$ unless there is built-in support for maintaining privacy.

It is not always clear what is the damage potential for exposing private information in a generalized presentation scenario. As noted before, some exposures only result in embarrassment or a temporary inconvenience for the presenter (search history, file system view), while other exposures can actually lead to a security threat or severe consequences (login name, confidential budget parameters, or exposing a recently used files list). The potential for either type of exposure often makes presenters feel insecure when going “on-the-air,” suggesting that the perceived (psychological) cost may be larger than the actual (logical) cost.

There is no doubt that allowing a presenter some control over what private information is exposed is important and can make a presenter feel more comfortable (Palen and Dourish 2003). Yet Hong’s risk assessment equation tells us that if the potential damage is not too high, expensive solutions are not likely to be adopted (e.g. collaborative-aware tools that require development, training and abandoning familiar end-user tools).

Many solutions will require presenter and viewers to install or purchase special tools. As Grudin (1988) pointed out, when the person enjoying the benefits of the technology (the presenter whose privacy is maintained) is not the person doing the work or suffering the costs (viewers that install and learn special tools bear the brunt) the technology will most likely fail.

The conclusion to draw from this is that for all these cases a simple solution that does not require completely new tools would be beneficial.

2.4.4 Private Information Sources in View Sharing

Lederer et al. (2003) suggest classifying private information types along two dimensions: Persona vs. Activity and Primary vs. Incidental Content.

In generalized presentations the first dimension is collapsed to mostly activity related information, as the audience already knows who is presenting and that he exists, so there is no anonymity.

Most sensitive information that may be revealed in such a session relates to past activities (favorite web sites, recent file menu items, search history or a view of a file listing) or to in-session activities (such as keying in a login name, handling an error dialog or altering network settings).

Many people use the same machine for personal purposes and work purposes and most of the tension arises when traces of activities from one domain appear in the other domain. Hawkey and Inkpen (2005) point out that whenever information that is not appropriate for the current view context is exposed there is tension. It does not have to be illicit information such as pornography (although this is often what many people think of as the canonical example). For instance, issues of confidentiality can also arise with proprietary or confidential business information that is visible on the shared view.

When considering possible control over the exposure of private information, it is important to understand its characteristics in regards to the other dimension (primary or incidental). The common types for the presentation scenarios are:

Semantic objects – Visual representations of objects in the document model and their attributes (a specific range of cells in a spreadsheet, a paragraph in a text document, or a dialog box showing properties). These can be considered primary content. Often, a presenter would like to specifically mark these objects as confidential or private, while still exposing the rest of the document (one of the difficulties is that an object may have more than one visual representation, even at the same time, such as in Figure 2.3b and 2.3c).

Peripheral data – Sensitive data that is not part of the object model of the shared

document but appears in the application's UI as a byproduct of the presenter's interactions (recent files, browser navigation history, auto-complete text boxes) and is therefore incidental to the activity.

Many applications have introduced personalized convenience features that cache users' preferences and selections and appear without explicit action on the part of the user. Other applications couple sensitive and non-sensitive UI controls (a global settings dialog that contains both Color and Security settings), so a presenter who tries to interact with control α will expose a sensitive control β on the path to α .

Interactions – Some of the interactions a presenter makes may be deemed private because they affect his reflected image, regardless of the data they operate on. These are mostly incidental disclosures. Some examples are committing syntax errors or other mistakes, searching for the right menu item, struggling with a wizard, or exhibiting slow typing skills.

Some exposures of private elements (primary or incidental) are an immediate outcome of the presenter's direct manipulations and fit well within the presenter's mental model of the application. These may be avoided or bypassed by the presenter at the price of forcing clumsier interactions or more careful preparation ahead of time. Other exposures are byproducts of agents that work on behalf of the user (e.g. an error message or the contents of an auto-complete text widget). These are less predictable and require more automated help to avoid accidental exposure. In either case it is disclosure to viewers we need to control, not the appearance or content of these elements. For example, it is possible to clear the contents of the navigation history or to filter it before using a browser in a presentation (as hinted by Hawkey), but this will take away important cues from the presenter.

Private information from any of the previously presented types that may appear in a shared view can be dealt with at several levels:

Task level – A presenter would need only to expose the windows and components that are part of the shared task, not all activity. This may entail sharing several applications or only one instance of an application (e.g. a single document). It is often not necessary or

desirable to expose the entire desktop (as seen in Figure 1.1).

Window level – An application usually comprises more than just a single window. There are dialog boxes, menus, palettes, toolbars and sub-window frames. In many cases these contain private information (file browsing dialog), they appear at awkward moments (error dialogs), or they just take up screen space. Clearly, not all of these components should be shared.

Visual Surface level – At the lowest level, we have the information bits visible on a single window’s surface, such as representations of underlying document objects or the contents of UI widgets. Our approach demonstrates how these can be dealt with as parts of the *image buffer* at this level.

When working on our system prototype, we realized that in many cases it is more effective to define a **state** of the application as private and freeze updates on the public copy until the application exits the state. This is useful when private data is mapped to externally inaccessible objects or associated with a large set of objects that cannot be treated individually (e.g. switching to a private worksheet or a show-comments mode or when an arbitrary error occurs).

It may seem that some of the private interactions described above are brief, so the amount of information viewers can extract is limited. However, it is common for shared sessions to be recorded, allowing later analysis so that ephemeral information becomes persistent (Palen and Dourish 2003). It has also been shown that viewers are quite likely to notice sensitive text on a large-screen public display, often used in co-located presentations (Tan and Czerwinski 2003), which suggests that in Hong’s equation the size of the display may increase L and probably the number of viewers in the audience will too!

2.5 Improving the Audience Experience

Recent work by Reeves, Benford, O’Malley, and Fraser (2005) has pointed out that interaction with computers is increasingly a public affair, such as interactions in museums and galleries or the growing use of mobile devices in public contexts. Therefore, there is a need

to consider not only an individual's dialogue with an interface but also to consider the ways in which interaction affects and is affected by spectators or "how should spectators experience a performer's interaction with a computer?".

In our first scenario passive viewers followed Bob's interactions. We can customize each person's view by adding, deleting, or modifying the application's presentation (bitmap) to provide a more useful experience, effectively optimizing the amount of salient information on the screen for each viewer.

2.5.1 Inadequacy of single-user GUIs for passive viewers

Passive viewers must follow the interactions performed by the presenter, but there is a perceptual gulf between the presenter and the viewers (Figure 2.4). While the presenter translates her intentions and semantic-level operations into GUI interactions, the passive viewers are doing the reverse process, inferring the underlying intentions and semantics from the interactions. This is not an easy process. Even when verbal explanations are provided by the presenter these are usually insufficient, inconsistent, and they require extra effort on the part of the presenter.

The process is similar to Norman's execution and evaluation gulfs, where the problem is simpler because the presenter is also the viewer (Norman 1988). This is illustrated in Figure 2.4, which depicts the viewer needing to deduce, based on the evidence provided by the interactions visible on the screen, what the presenter is doing and why she is doing it.

One of the root problems is that the visual language of most existing graphical user interfaces is highly tuned for a single active user, ignoring the needs of passive viewers. For example, when a presenter decides to perform a contextual menu selection, the cue for a passive viewer that some interaction is about to take place is the appearance of the menu, by which time it already obscures most of the context for the operation (Figure 2.5a).

Another problem is that passive viewers tend to follow the presenter's point of interaction (often highlighted in GUIs), yet in some cases the presenter would like to draw attention to other regions of the display.

Some operations have no feedthrough (the feedback produced when artifacts are manipulated) or feedthrough that is inadequate for passive viewers who will simply miss it, as can happen with collaborative-aware groupware solutions (Gutwin and Greenberg 1998a). This implies that viewers will miss critical clues that assist them in bridging over the deduction

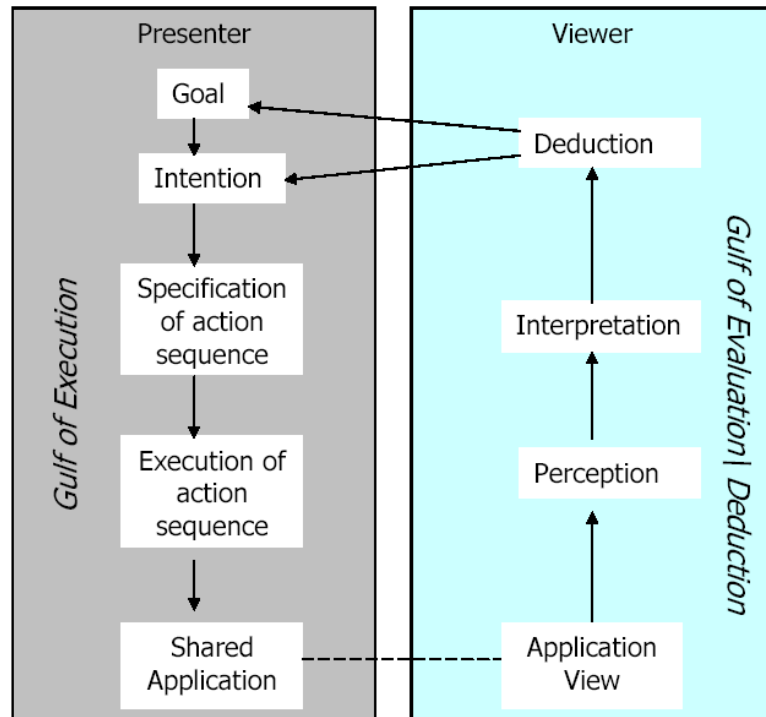


Figure 2.4: The presenter translates his goals and intentions to actions on the shared application (execution), whereas the viewer has to interpret the presenter's goal and intentions from the reflection of the actions on the shared view (interpretation and deduction). Adapted from Norman's model. In the original model the presenter and the viewer are the same person, so the problem is much simpler. The presenter interprets the state of the system to **evaluate** the results of his actions rather than deduce his original goals.

gulf (Figure 2.3).

In Figure 2.5c, the presenter was using a keyboard shortcut to move between two states of the text. Viewers have no way to tell what command was used because there is no feedthrough as there would be had a menu selection been made.

Other low-level parameters such as cursor size or shape, the time a menu or dialog remains on-screen after release, and the way selection highlighting is done, are tuned for the performance of a single active user. These are not suitable for passive viewers, who are trying to follow the interactions without the benefit of knowing the intention of the action or experiencing the kinesthetic feedback of mouse or keyboard interaction.

The shape and size of the mouse cursor are a particularly interesting example. Po, Fisher, and Booth (2005) remark that modern GUIs are still using more or less the same original cursors that came about in the mid 1970's. These cursors were partially crafted

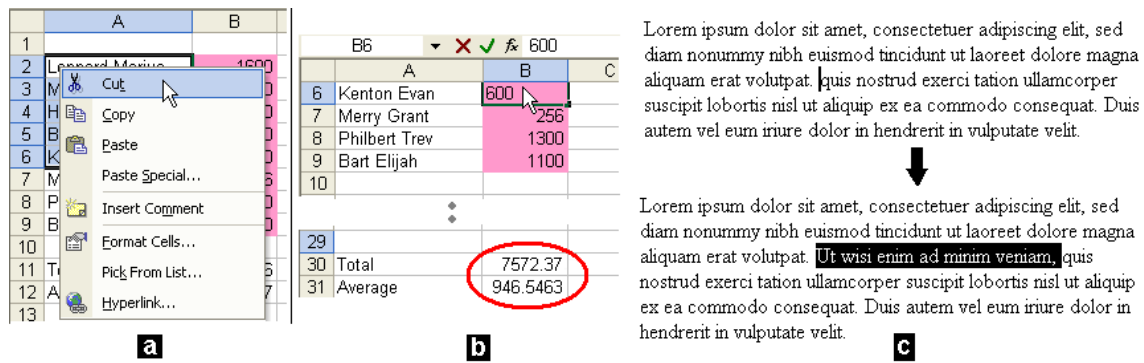


Figure 2.5: Inadequacy of conventional GUI for passive viewers: (a) a menu obscures the context for the operation; (b) viewer focus follows the mouse interactions with the top cells, but the presenter wants viewers to focus on the bottom cells that are changed as a result of the interaction; and (c) a non-visible keyboard shortcut was used to insert text (for instance undoing a delete operation), so viewers cannot tell what caused the change.

based on the hardware constraints of the time. They were purposely designed to be not too small for the user to see yet not too large so as to obscure too much of the display. To the best of our knowledge, no real evaluation has ever been conducted on their suitability for passive viewers.

As we will show in Chapter 4, some of these parameters can be independently changed on the public view of an application to better fit the needs of passive viewers in the audience.

2.5.2 Controlling verbosity

Viewers may have different levels of expertise and familiarity with a shared application. It is beneficial to adapt their views to the appropriate levels. A key aspect to be controlled is the *verbosity* of interactions or the amount and level of interaction details. For example, if Bob is to teach Ted and Alice how to fill out a report using an application unfamiliar to them, exposing the fine details of his actions (menu selections, dialog boxes, etc.) and adding cues (like keyboard shortcuts and change highlighting) could be crucial.

On the other hand, if Ted and Alice were experienced users, exposing each and every interaction or adding too many cues will prevent them from concentrating on the semantics of the report.

From a pedagogical point of view, it sometimes makes more sense to show one logical interaction unit as a single visual step, so the high-level semantics are not obscured by the

low-level details.

Reeves et al. (2005) propose to control the representation of both the manipulations a presenter makes and the effect of these manipulations in four main levels: hidden, partially revealed, revealed and amplified. They then present a taxonomy of possible spectator views, adapted here for our scenarios:

- “Secretive” - Both interactions and effects a presenter makes are hidden. This is useful for private components or for interactions that are irrelevant and could only distract viewers.
- “Expressive” - Interactions and effects are revealed or amplified. This is useful for teaching scenarios, making sure viewers understand how to accomplish a certain task.
- “Magical” - revealed or amplified effect, while interactions are hidden. Useful when presenting to expert users who do not care how the interaction was made or when display space is limited. In both cases it is still important for viewers to stay alert for content changes.
- “Suspenseful” - revealing and amplifying interactions while hiding effects. Useful when interacting with private content. The actual changes made to the content cannot be seen by viewers. Therefore, they need to compensate by getting better understanding of what actions were performed.

2.5.3 Mitigating visual clutter

We have already implied that it is desirable to share only relevant windows or components, rather than the entire desktop. This can assist viewers in making better use of their screen space. This is especially true for remote viewers who work with other applications on their screen in parallel, or for multiple co-located users who bring up applications on a shared display as in Figure 2.2.

In the case when all sub-windows, dialog boxes and menus are automatically shared as well, viewer’s display can quickly become cluttered. This is somewhat like violating acoustical privacy with cellular phones (Palen and Dourish 2003). The presenter imposes his “conversation” with the application on the viewers, much like a person talking on a cellular phone imposes on others in a public area.

A viewer should be capable of controlling how much of this conversation penetrates his display and replace some interactions with other “low volume” representations.

In all cases viewers need to maintain some level of awareness of the presenter’s actions because this has been shown to be important for collaborative work (Gutwin and Greenberg 1998b). However, it does not always have to be in a one-to-one manner. Alternative representations may be more effective for passive viewers. Moreover, when bringing in privacy concerns it is clear that the awareness level should also be balanced with privacy constraints. Fortunately, these two goals are not necessarily in conflict. Removing private information reduces screen clutter and thus increases awareness of the elements that remain visible.

Chapter 3

Related Work and Literature

There is a large corpus of research on tools that support collaborative work and that can provide varying degrees of application view sharing. However some are targeted at shared editing and are not suitable for the presentation scenarios in which we are interested. Other tools are suitable for presentations, but cannot meet the privacy and augmentation demands. In this chapter we survey the relevant previous work, highlighting the features and approaches that best fit the requirements we identified in the previous chapter.

3.1 Collaboration-Aware Solutions

Many collaboration-aware tools and frameworks that allow the creation of custom view sharing with varying synchronization degrees are reported in the literature. There have been a number of surveys of other systems and toolkits (Greenberg and Roseman 1999; Begole, Rosson, and Shaffer 1999). There are also commercial tools that support collaborative writing and flexible viewing of documents, for instance SubEthaEdit¹.

GroupKit (Roseman and Greenberg 1996) allows easy coding of real-time distributed multi-point conferences between users. Using its collection of multi-user groupware widgets it is easy to create relaxed WYSIWIS file viewers. Gutwin and Greenberg (1998a) suggested enhancements to the toolkit with custom controls that assist passive viewers. Some of these are illustrated in Figure 3.1.

The privacy concerns that we are interested in (restricting view) are typically not addressed by these systems. Most systems are designed for true synchronous collaborative work and focus on providing access control as a means to regulate privacy (see a survey by Tolone, Ahn, Pai, and Hong (2005)).

Managing a collaborative session creates a substantial amount of overhead, much like managing one's privacy or providing proper awareness for passive viewers. Some work

¹<http://www.codingmonkeys.de/subethaedit/> (last checked August 8th, 2005)

has looked into providing rule and policy based automated regulation of collaborative sessions (Edwards 1996). Lau, Etzioni, and Weld (1999) looked at rules controlling what pages are saved into the web browsing history and created a simple “privacy policy editor interface” for managing them. Such policies regulate reactions to events and system states that reduce the unpredictability of the system and require less management efforts from the user.

Recently, some effort has been targeted at managing sensitive transactional information that may be exchanged in collaboration settings, notably the work in the Platform for Privacy Preferences (P3P) Project².

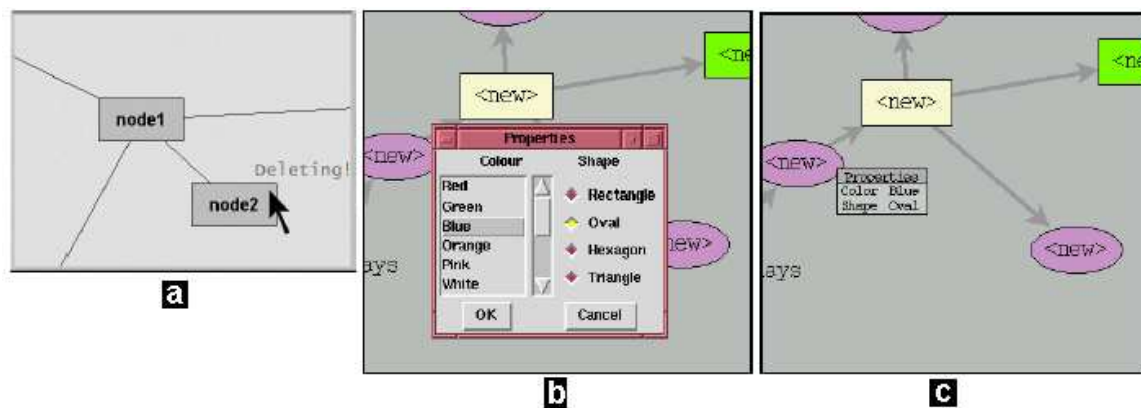


Figure 3.1: Some custom controls can be developed using GroupKit that assist passive viewers in following a person who is interacting with the groupware. (a) Providing visual feedback on a delete operation. (b) A dialog that opens on the presenter screen, shows only as mini-summary on the viewer screen (c).

However, many of these tools are usually not acceptable solutions in the real-world. Most of them were only built as proof of concept or as laboratory tools with a minimal feature set that cannot match that of commercial end-user tools. Moreover, Li and Li (2002) note that groupware features are used less frequently than features supporting individual activities, so being forced to learn new interfaces for a sporadic task, such as occasional view-sharing, will discourage presenter and viewers from using such solutions. Rather than retrofit all of the features present in existing single-user applications to the new collaboration-aware frameworks and toolkits, it seems more sensible to extend the single-user applications to support collaboration when that is possible.

²<http://www.w3.org/P3P/> (last checked August 8th, 2005)

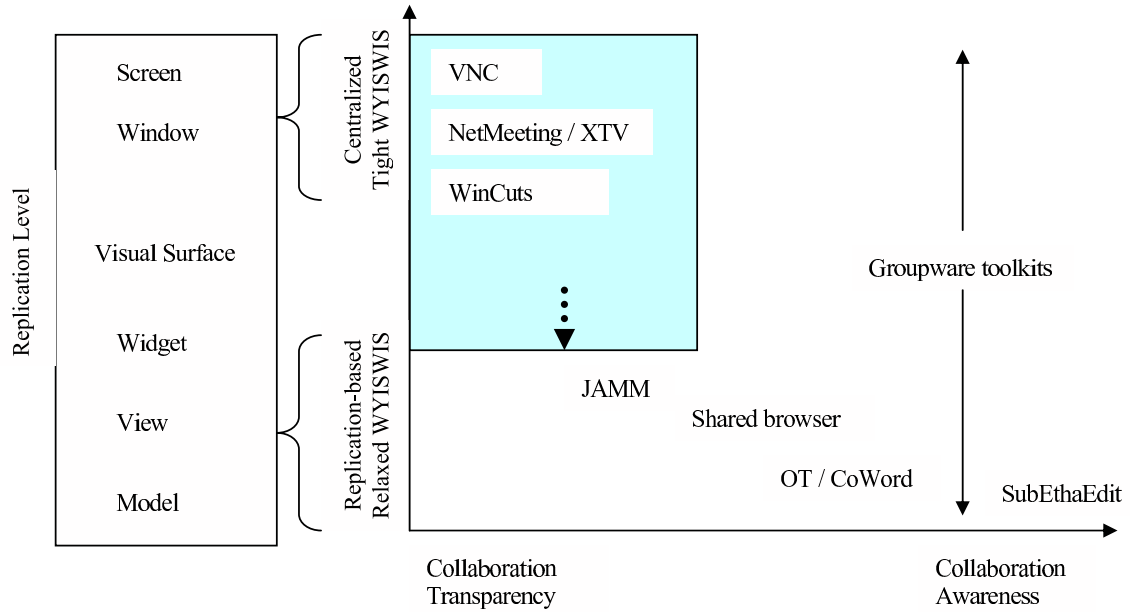


Figure 3.2: The replication level (based on the Zipper model) affects the amount of WYSIWIS relaxation, but also the synchronization effort. Collaboration-aware solutions and groupware toolkits offer richer options of decoupled views, but are very specialized. On the other hand transparent solutions are general but offer limited ability to relax views (being centralized and bounded by the window level). The cyan colored rectangle shows where our solution fits in. By allowing some reliance on application and windowing semantics one can farther relax WYSIWIS into the visual surface level and support privacy and passive viewers.

3.2 Collaboration-Transparent Solutions

Our system is targeted at existing collaboration transparent single-user tools that were not designed for multiple views and cannot accommodate code changes. The lack of access to source code is but one reason why code changes might not be possible. A prominent example of this type of software is the collection of off-the-shelf desktop tools such as text editors and spreadsheets that enjoy widespread use by single users and which form the basis for many of the *ad hoc* collaborations that follow the generalized presentation model we wish to support.

In this context, it is useful to classify these based on a mapping of the sharing architecture space. The Zipper model, presented by Dewan (1999), looks at the common layers that can be shared: Screen, Application, Window, Widget, View, Model and classifies the solution architecture based on the share branching level. In each architecture one of these layers is the branching point. All layers below it are shared and all layers above it are

replicated and therefore influence the level of WYSIWIS relaxation.

Figure 3.2, shows some of the possible solutions (with varying degrees of collaboration-awareness and replication level)

3.2.1 Centralized tools

VNC and NetMeeting - Closest to our approach are tools that replicate the screen or window layers (known as centralized tools), such as VNC (Richardson et al. 1998) or NetMeeting³. XTV (Abdel-Wahab and Feit 1991) is an equivalent tool (it replicates X-Windows display commands instead of replicating a bitmap).

VNC shares the entire screen, including windows of applications a presenter might like to keep private. NetMeeting and XTV share all windows of a chosen application, and they lack the ability to keep some of the dialogs or palettes private. NetMeeting and similar solutions allow sharing of a fixed region of the screen, but if windows are moved or resized outside of the region bounds they will not be shared. Worse yet, if an embarrassing error dialog or new window that should be private pops up in the shared region, it will be shared.

In terms of supporting different views, NetMeeting and its successor, LiveMeeting, allow a presenter to manually “pause and play” the view sharing (Figure 3.3). This requires the presenter to identify privacy concerns, some of which are unpredictable, while he is “on-the-air” and is therefore error-prone. For example, a presenter who is about to type into an HTML form field would have to anticipate that an auto-complete box will appear and then “pause” updates beforehand. The presenter would also have to remember to press “play” after the auto-complete box has gone away. Manually managing these incidents is quite cumbersome. The presenter should be able to specify rules for controlling these private elements in advance.

All of these centralized tools lack any ability to systematically change the contents of the replicated screen parts and therefore cannot handle sub-window elements, protect private content within an element, or provide highlighting to assist viewers (some tools allow manual highlighting or sketching on the application surface, but that becomes invalid once the window is scrolled or resized).

³<http://www.microsoft.com/windows/netmeeting/>. There is a plethora of commercial tools that offer similar functionalities such as: Bridgit (<http://www.smarttech.com/support/product/bridgit/>), webex (<http://www.webex.com>) or SunForum(<http://www.sun.com/desktop/products/software/sunforum/>) (all web sites last checked August 8th, 2005)

More recent research has looked at using similar “screen-scraping” tools to promote group awareness by publishing live bitmaps of participants’ displays (often scaled down). The Notification Collage (Greenberg and Rounding 2001) supported the posting of a live desktop image on a publicly accessible display so that lab-mates can get a notion of what their colleagues are doing, and whether the colleagues are available or need assistance (Figure 3.4b). Commercial remote desktop solutions allow the viewing of multiple remote desktops mostly for remote assistance purposes⁴. It is clear that all of these tools have severe privacy implications. Moreover, when observing a desktop in miniature, especially for remote assistance purposes (Figure 3.4a), it is easy to miss critical interactions and details that may require alternative representations or visual augmentations that suit a smaller view.

Despite all of the drawbacks, centralized bitmap-based sharing tools are still very popular due to their high degree of collaboration transparency and the low demands on the part of the viewer (just a thin client is required by the viewers because most of the work is performed on the presenter’s computer). This is a property our solution tries to maintain.

3.2.2 Replication-based tools

Many collaboration supporting tools are based on the notion of replication. Each person runs his own copy of the shared application. Solutions differ in the way they synchronize these two copies:

Flexible JAMM - This is a framework for transparently replicating the widget layer of Java Applets at run time (Begole, Rosson, and Shaffer 1999). It is based on replacing some widgets and components of existing single-user applications with multi-user versions that can be synchronized with some degree of differentiation and provide cues on the actions of the other users. The focus of this approach is on location-relaxed WYSIWIS (users can look at different parts of a document).

While aimed at preserving collaboration-transparency, this approach puts different constraints on the running environment and underlying code (e.g. it relies on specific Java Swing features and the use of a custom class loader). Cheng, Rohall, Patterson, Ross, and Hupfer (2004) proposed using aspects and pointcuts to transparently add collaborative capabilities to shared applications, mostly for awareness purposes. However, neither tech-

⁴<http://www.apple.com/remotedesktop/> (last checked August 8th, 2005)

nique easily addresses privacy concerns and both are unsuitable for commercial off-the-shelf applications that do not expose their code.

Operational Transformations - Recently some research efforts have focused on synchronizing two running single-user applications (i.e. replicating the Model layer), using operational transformation (OT) techniques. This approach allows users to use the tools they are familiar with in a collaborative setting, but at the expense of blending in some application specific semantics (Li and Li 2002). OT is largely aimed at keeping two copies of a document synchronized by capturing user editing operations and sending a transformed version of them to the other copy. As such these techniques can be used for the generalized presentation scenarios, where presenter and viewers each have a copy of the document.

A prominent example, working with an off-the-shelf editor (MS Word) is CoWord (Xia, Sun, Sun, Chen, and Shen 2004). Each user has independent control of her copy while the system synchronizes the underlying document models using the application's API. Co-Word does more than we need. It allows modifications to be made freely by all of the participants, and keeps track of conflicts when they arise and either resolves them or reports them to the participants. The generalized presentation scenarios we want to support assume that only serialized changes are made by different participants, and most often only a single participant (the presenter) makes modifications.

These solutions assume complete independence of views, which is not suitable for presentation scenarios, and typically they do not provide any privacy protection. Still, with some effort views can be synchronized as well (replicating the View layer) and OT can be extended so that private document parts are converted or changed on the viewers' copy. Because each participant is running a different copy of the application there is less opportunity for incidental privacy leaks, but not entirely. There is implicit shared information propagated by the synchronization protocols that may reveal information that was not intended to be shared in the document, such as intermediate text resulting from a cut-and-paste operation that is modified in the document but can be seen in its original by other users through change-tracking mechanisms.

There are similar solutions that work with other software tools. For example Sakairi, Shinozaki, and Kobayashi (1998) devised shared browsing that also synchronizes HTML

form fields, unlike simple shared browsers that just point at the same URL. Also worth noting is a commercial tool that can synchronize Excel versions.⁵

The major drawback of JAMM, CoWord and other replication-based solutions is that both presenter and viewer need a copy of the application, or alternatively two instances running on a single computer. This is often a harsh demand that cannot be met (e.g. viewers do not have a license to run the application, they have an incompatible version of it or the presenter's machine cannot run two copies of the application).

Furthermore, synchronizing the application replications and the views can be a hard problem (Dewan 1999). It often requires resource locking or forcing expensive calculations to be carried out multiple times and even then there are still inherent conflicts that must be resolved in some manner when more than one participant modifies the same part of a document or object.

3.3 Screen Recording tools

There are several tools that allow recording a computer's display and basic editing and indexing of these recordings.

Recording

Most tools rely on the same techniques that the centralized tools presented beforehand use to capture screen image buffers. Yet, these tools add an additional component that encodes these images into a video sequence. The `vncrec`⁶ tool is a basic screen capturing tool that simply records the broadcast messages of the VNC protocol and then replays them in order. Commercial tools like Camtasia also provide manual editing capabilities on recorded videos (deleting interaction sequences, drawing and adding annotations or choosing screen regions to magnify) and some automated augmentation effects (adding a visual indication on mouse clicks).

⁵A commercial tool enables multiple users to work collaboratively on an Excel worksheet in real-time (<http://www.advancedreality.com>, Last checked August 17 2005).

⁶<http://www.sodan.org/~penny/vncrec/> (last checked August 18th, 2005)

Indexing

The ability to search recorded screen interactions can be very useful. One simple solution (Li, Spiteri, Bates, and Hopper 2000) relies on the lower level VNC protocol events as indexes to search for certain keystroke combinations or for updates that changed at least 40% of the screen. It is hard to relate this level of indexing to application interaction events.

Lately several observation and web-usability testing tools⁷ have incorporated similar indexing capabilities on screen recordings with some additional data channels: capturing window events (open,close,focus), screen text or web browser page changes and physiological data (such as heart rate or eye-tracking).

3.4 Spatial and Window Set Manipulations

Wincuts (Tan, Meyers, and Czerwinski 2004) is a collaboration transparent bitmap-based window sharing system that provides some spatial manipulation of shared windows. It allows a presenter to manually select a region of the window and publish only that part (see Figure 2.2).

Similar ideas are presented in regards to window layouts and screen space use for window managers by Hutchings and Stasko (2004). In many cases only the relevant part of a window should be kept visible or shared.

While still allowing a lot of flexibility and addressing clutter and privacy issues, this approach quickly breaks down when a presenter needs to resize or scroll a window. As we shall explain later, our system completely subsumes these approaches, automating spatial manipulations and blending them with other filters.

Subsequent work by Hutchings and Stasko (2005) is also relevant to the manipulation of an application's window set. Dialog and notification windows are classified using OS calls and duplicated on multiple displays so a user can easily spot them. Similar techniques can be used to determine which dialogs and windows should be duplicated on a public display and which windows should remain on the presenter's display.

⁷Two examples are Morae (www.techsmith.com) and The Observer[®] (www.noldus.com).

3.5 Visual manipulations

Several techniques for systematically modifying the visual surface of an application were explored in past research:

Surface Manipulations - An interesting set of manipulations to existing applications' visual surfaces was presented by Olsen et al. (1999) and Edwards et al. (1997). The visual manipulations are aimed at supporting the work of a single user (allowing text search and search result highlighting, radar views and visual bookmarks).

On the technical side, this approach requires overriding some of the low-level drawing routines of the graphics object (or drawable object in the subArctic⁸ toolkit they used). It also relies on consistent ordering and grouping of component drawings and adding keyword hints at certain points in the code (Figure 3.5). Given these hints, when the application renders itself on the drawable object it is possible to extract locations of objects and text on the visual surface and add custom filters.

This approach resembles the idea of a “magic lens” (Bier et al. 1993), that replaces PostScript commands to create an alternative view of the visual surface beneath it. This type of approach cannot reason about information that does not go through the display pipeline (like field names). Some of the technical demands (adding special grouping calls) are not fully met by off-the-shelf tools and are quite hard to support under some architectures (for instance it is not easy to override the low level drawing routines in Windows).

Still, this may be a viable way to perform some of the “semantic glue” operations and visual manipulations that we will discuss in Section 4.1.2. The main advantage of this technique is that it is efficiently blended into the drawing pipeline and therefore locations extracted on the visual surface are guaranteed to be synchronized with the currently visible objects.

Our system employs similar filters that can be used in a multi-user shared view scenario. We will describe alternative channels of information to allow graphic parsing of the visual surface using a plug-in architecture that is part of our system in Chapter 4. The plug-ins allow us to incorporate these techniques into our system.

⁸<http://www.cc.gatech.edu/gvu/ui/sub-arctic> (last checked August 18th, 2005)

Blur filters - Blur filters have long been used for privacy purposes in video recordings and conferencing. Boyle and Greenberg (2005) survey some basic automated techniques to detect states where the video contains private information. However, accurately processing a video signal is not always possible and constantly applying some level of blur to conceal private details will result in loss of awareness (Neustaedter, Greenberg, and Michael 2005). In contrast, by parsing the visual surface of a shared application (which we will demonstrate later), it is possible to apply selective blur that does not impede awareness but does protect privacy.

Another interesting use of blur filters was introduced by Blackwell, Jansen, and Marriott (2000). Blurring is applied to an image of an application, apart from a fixed size region that the user can move using the mouse. The user's gaze is "coaxed" to be on this non-blurred part. Thus, knowing where the non-blurred region is (mouse position) tells us what the user is looking at.

This scheme demonstrates that blurring operations not only serve as privacy preservers, but can also provide helping cues and direct attention of passive viewers to important screen parts.

Subjunctive UIs - Some work has been focused on simultaneously showing parallel system states to reveal the outcomes of all combinations of a specified set of parameter values. Theoretically, some of the parameters can be used to regulate privacy or augmentation and provide two different views of the application, one for the presenter and the other for the audience. These are subjunctive because they show the results of future actions that may or may not be taken, in effect allowing the user to see if the desired effect will be achieved before committing to the action.

An interesting example is Side Views (Terry and Mynatt 2002) that augments a few existing open-source applications in a semi-transparent way to allow live previews of command actions. It drives the original application as a computation server on a copy of the document. The desired commands are then executed using the open-source code (such as applying bold typeface to a line of text or an image filter). This approach relies on tight integration with the underlying application and in many cases requires expensive computations to produce the alternate views.

3.6 Multi-Machine User Interfaces solutions

In regards to privacy and clutter, Pebbles (Myers, Peck, Nichols, Kong, and Miller 2001) replicates some application components on a handheld device. Thus a presenter may choose to conduct some interactions on her handheld or auxiliary computer to avoid exposure. Greenberg, Boyle, and Laberg (1999) proposed a similar solution.

This approach requires extra hardware to be present and it does not address viewer needs for awareness cues. Most importantly this approach is limited in the type and complexity of components that can be recreated on the handheld (text fields and menus work well, but part of a worksheet relying on other spreadsheet parts may not). In the end, it is equivalent to the two-display extended desktop situation.

Other systems rely on the availability of PDAs and laptops for audience members instead so that personalized views of a presentation can be displayed on them.

Hexel, Johnson, Kummerfeld, and Quigley (2004) developed a solution that personalizes Impress (OpenOffice.org) or PowerPoint® slides. An important component of the system is a context manager that exploits the personal devices as a channel for obtaining audience member profiles. These profiles form a basis for real-time altering of content presented on individual and room displays.

The system relies on slides being represented as XML information that can be altered based on rules (much like personalized web pages are produced) and on off-line authoring of alternate slides.

For example, some viewers will get additional subtitles with German translation, but the slides on the public display may be altered so private information will be omitted from them while still being available for certain audience members.

The ideas of a context manager that can drive multiple views is very compelling. It is not clear how this solution can apply to other applications that do not provide easy access to the displayed content or to GUI components that are exposed while interacting with the presentation.

3.7 Single Display Privacyware

Research into single display privacyware has resulted in several platforms and hardware setups that enable multiple users to each have a different view of a shared display by using

shutter-glasses (Shoemaker and Inkpen 2001; Yerazunis and Carbone 2001). For example, they enable one user to bring up private information on the shared display without the others seeing it. These solutions require additional hardware (LCD shutter glasses) and are not fully secure because it is easy to circumvent the privacy safeguards by removing the glasses.

Moreover, these systems still require running software that is capable of supporting differentiated views, so collaboration-aware tools or special software is used. The solution we propose can allow existing application to enjoy some of the advantages of such setups without requiring special hardware or collaboration-aware tools.

3.8 Presentation tools

Perhaps the best existing solutions for our requirements are, not surprisingly, tools developed specifically to support presentations. Recently, commercial presentation authoring and playback tools such as Microsoft PowerPoint^{®9} and Apple Keynote¹⁰ have taken advantage of multi-display technology to play the presentation on a public screen, while providing a private view to the presenter on her laptop (where she can view her notes or check other slides). The drawback is that these work only within the application, not across multiple applications that are used within a generalized presentation. Our approach can provide similar advantages to other single-user applications and they can be used together in a generalized presentation.

3.9 Presentation Authoring

The idea of automating the design of the graphical presentation of information, interfaces, multimedia content and presentation slides to maximize their effectiveness for an active user or for viewers has been explored from many different angles.

Searching the Design Space

A pioneering work was APT (A Presentation Tool) by Mackinlay (1986). APT was designed to automate the design of the graphical presentation of relational information (charts, plots

⁹<http://office.microsoft.com/en-us/assistance/HP030893931033.aspx> (last checked August 18th, 2005)

¹⁰<http://www.apple.com/iwork/keynote/presenter.html> (last checked August 18th, 2005)

and graphs) by applying a search architecture on the design space.

The fundamental idea is that graphical presentations are sentences in a *graphical language* and are composed of primitive graphical techniques. Thus, a search architecture can examine the space of these building blocks, taking into account the nature of the data, user preferences and the properties of the output medium and tailor a presentation that will maximize the presentation effectiveness. Effectiveness was mostly evaluated on how accurately people perceived the generated design.

A similar problem arises when considering possible modifications and enhancements that can be applied to the shared view of an application. In Chapter 4 we will describe a set (or language atoms) of visual manipulations that can address privacy concerns or assist passive viewers. To make the shared view more effective an automated search of this set is required.

Applying AI techniques

Another approach to tailor information presentation to the needs of viewers relies on using user models and AI reasoning in the authoring process.

For instance the Valhala system (Csinger, Booth, and Poole 1994) introduced “intent-based authoring” in the domain of video authoring. Form and content of edited video sequences and video annotations are determined based on the author’s (presenter’s) intent and a model of the viewer and his needs.

Viewer models can be constructed explicitly (viewer choices) or implicitly by observing viewer actions and interactions and inferring his characteristics.

Another important input for the authoring process is the media characteristics. Each media format (video vs. paper or workstation screen vs. large screen display) has its own limitations and advantages that an authoring process should consider.

It is possible to use similar techniques for editing a shared application view. The viewer model can incorporate knowledge about expertise with the shared application and privacy restrictions. Presenter intent could be formalized as well (e.g. train the viewer on the application vs. update the viewer on recent budget changes that happen to be in a shared spreadsheet). Intent-based authoring can then transform the view using various visual manipulation techniques and annotation techniques according to these models.

3.10 Animation

Animation has been studied in research as a means to improve user understanding of an application (usually focused on the active user and not on passive viewers). Early work by Baecker and Small (1990) studied potential uses of “animation at the interface”. An important aspect of this work looked at the use of animation for making an interface more comprehensible. Baecker demonstrated how animation can show a user what has been done, answering a question like “how did I get here ?” or convey transitions between application states and cue the user on new and old areas of interest. This early work also demonstrated effective use of animated icons to provide feedback on a system’s state or to intensify menu selections.

Thomas and Calder (2001) and others examined different cartoon-style animation effects for augmenting direct manipulation of UI elements in custom UI toolkits. Effects like squash and stretch, motion blur and dissolves were implemented.

One of the initial examples were pulldown menus. These were modified so that the a menu gradually expands (“slow-in”/“slow-out”) from zero to full size over a short interval of time.

Other animation techniques were applied to indirect manipulation effects (e.g. an align objects command) that require cognitive re-parsing of the new visual state. In such a case, animation that simulates a drag-and-drop operation can be helpful.

The challenge is applying animation techniques in a transparent fashion to existing applications so they can then be used to assist passive viewers in the context of generalized presentations.

"Play", "Pause" and "Stop" controls

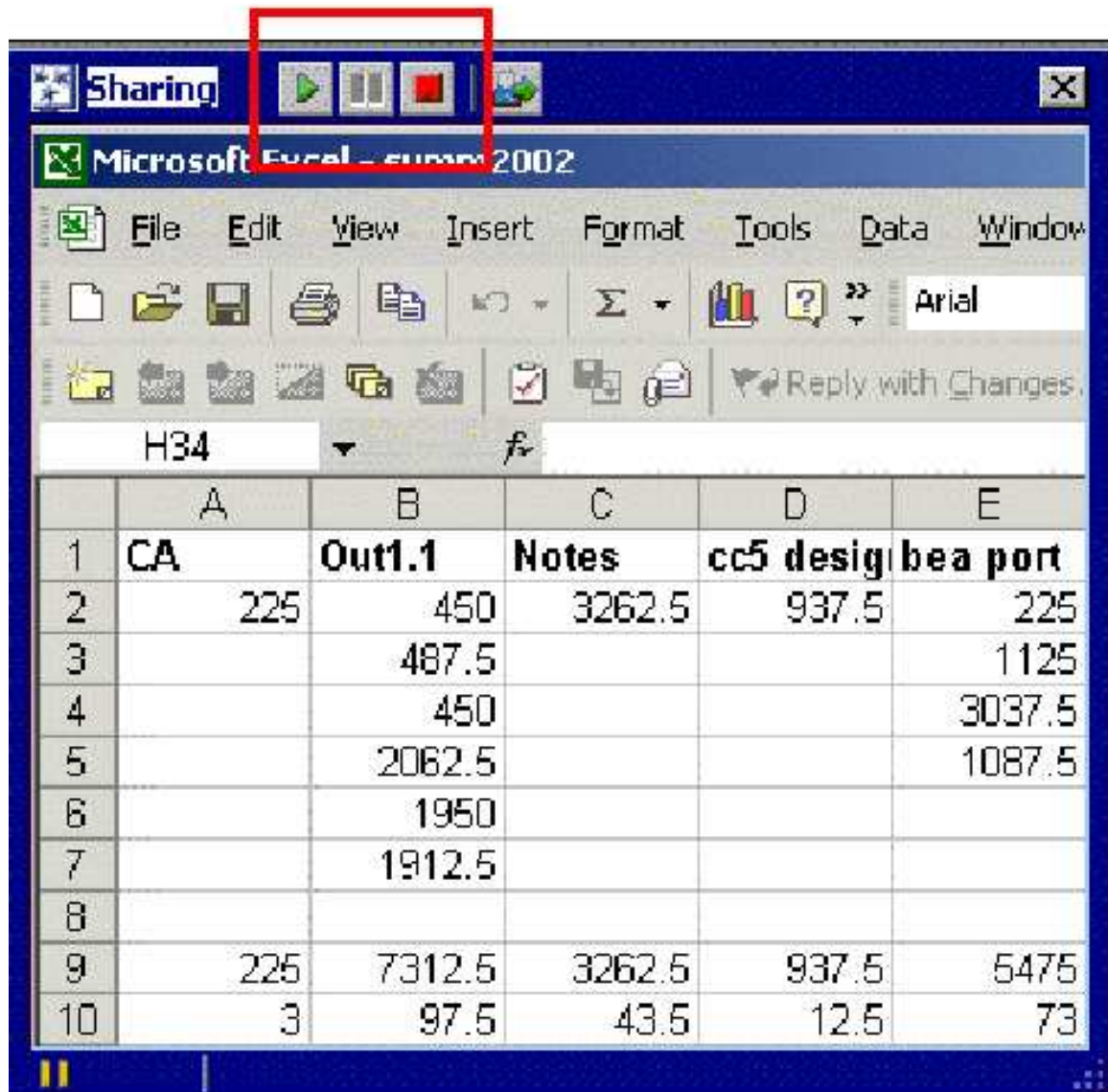


Figure 3.3: LiveMeeting is used to share a region of the screen containing a portion of a spreadsheet. The presenter's control is limited to manually pressing the pause and play buttons before engaging in a private activity, such as using a "File Open" dialog.



Figure 3.4: Sharing screens to promote awareness: (a) a remote desktop viewer observing a group of people for potential assistance in a classroom scenario; (b) some group members have posted mini-views of their desktops (marked in red) on a public screen using the Notification Collage.

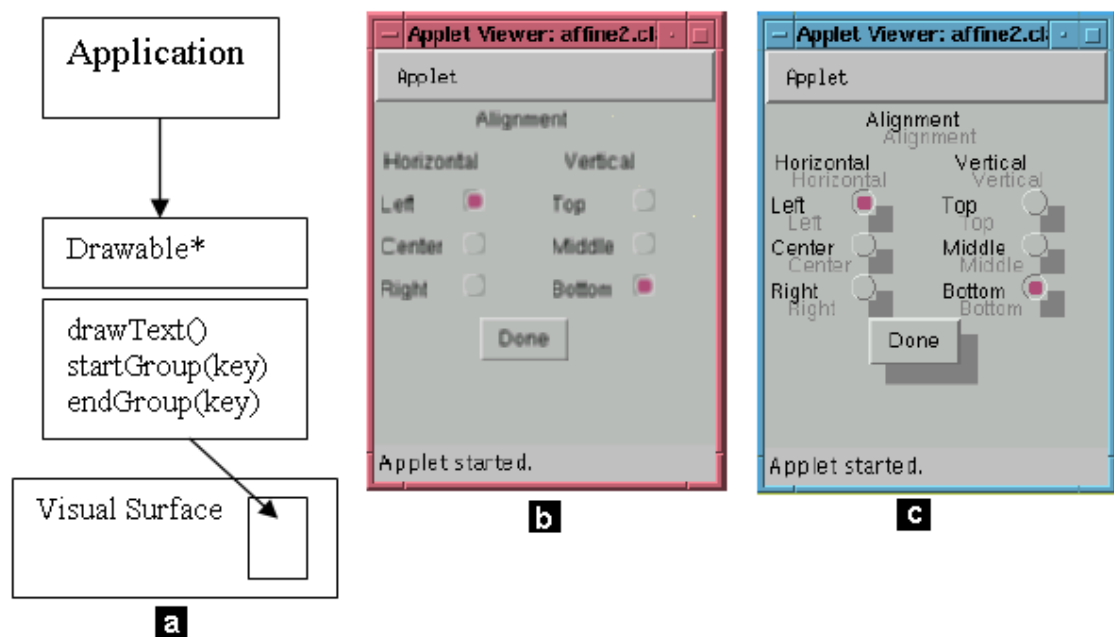


Figure 3.5: Visual surface manipulations: (a) the application is forced to render into a special `drawable*` object (inherited from the regular `drawable` / `graphics` object); `drawText` calls are captured as well as special `startGroup` and `endGroup` calls that are planted into the application's code to indicate where objects are on the visual surface when it is their turn to be rendered; (b) a blur operation is applied to UI controls; and (c) a highlight / shadow operation

Chapter 4

System Description

We have implemented a prototype of the system using C# under Windows and tested it with three widely-used commercial applications (Microsoft Excel[®], Word[®], and Internet Explorer). For the prototype we rely on a variety of compatible features in the applications, but the principles apply to any modern operating system and scripting language. They should work with any application, although some additional “semantic glue” layers that we describe later may be required in the most general case.

Most of the functionality described in this chapter was implemented or partially implemented in the prototype. Functionality that was not implemented and possible improvements will be indicated where appropriate.

4.1 Core Functionality and Components

Our system relies on a number of support functions that comprise the overall architecture. We describe the support functions. In the next section we describe how they are used.

4.1.1 Cloning Windows

In order to support differentiated views, our system grabs the visual surface of shared application windows on the presenter’s machine and conveys a manipulated version of the bitmap to the public display, published in *clone windows*. This functionality is provided by the Frame Buffer Grabber module (Figure 4.1).

In the prototype, we simply relied on timer-based device context copying, similar to Wincuts from Microsoft Research (MSR) (Tan et al. 2004). This approach matched our initial focus on co-located scenarios, where all displays and views are controlled from a single machine.

A more efficient solution might use a modified version of the Remote Frame Buffer (RFB) protocol (Richardson et al. 1998), adapted to work on separate windows. The RFB

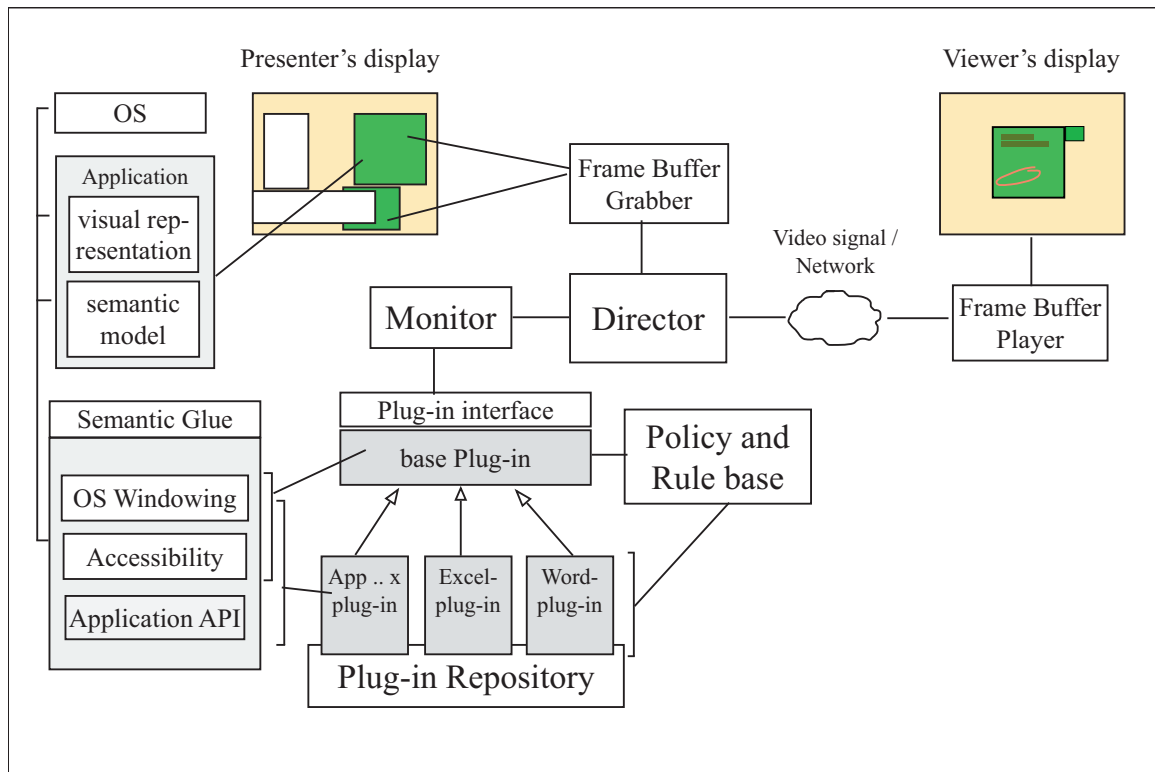


Figure 4.1: System Architecture – the main components of the system. See text for descriptions. The image buffers from the presenter's display are captured and manipulated using the semantic glue queries before being transferred (over video or network channels) to the viewer's display.

protocol uses paint events to trigger copying of changed parts only.

In both solutions the viewer client is a simple image buffer player (Figure 4.1, right) that is completely independent from the shared application.

We use extended desktop mode to control the presenter and viewers' displays. In this mode the public display is a logical continuation of the presenter's desktop, although often physically located on a wall behind the presenter (Figure 4.2).

A clone of each of the shared application's windows is created by querying the system's list of windows and making bitmap copies of the parts that are shared. The clones are automatically placed on the part of the desktop lying on the public display, and they are updated as the application modifies the originals.

The presenter can move any of the application's windows on her display or cover them with other windows without affecting the published clones.¹ The novelty is in how we

¹In modern operating systems each window renders itself to a separate graphics object, so even when windows are covered it is still possible to grab only the specific window's visual surface. Under Windows,

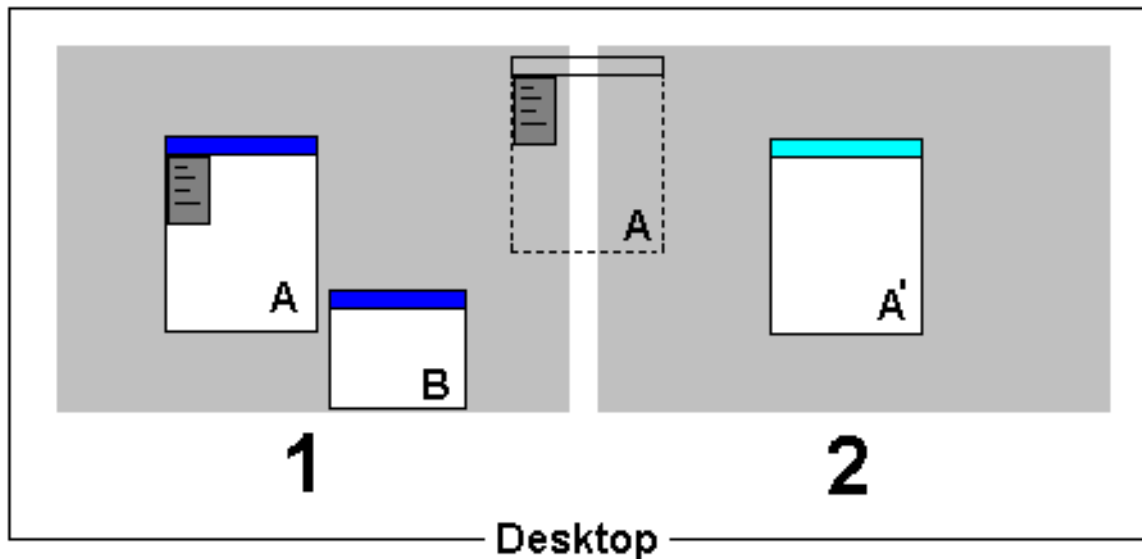


Figure 4.2: The presenter’s laptop drives both displays. Display 2 (public display) is not a copy of Display 1 (laptop screen), which is the the common mode for doing presentations. Instead the presenter’s desktop spans the two displays. The presenter could drag the shared application A to the public part of her desktop, so application B can remain private. However, if application A has a private component visible (e.g. a menu) the presenter will have to drag the window back and forth between the displays, conducting private interactions on Display 1. Furthermore, it will be hard to control on which display private popup windows appear because most applications are not fully designed to work in multi-display environments and popup windows often follow the main application window. In our solution, application A resides on Display 1 and a clone A’ is published on Display 2. Our system guarantees that private elements visible in application A will not get copied into A’.

modify the bitmap images and window set before they are placed on the public display.

4.1.2 “Semantic glue” queries

To alter the shared view along the lines discussed in previous sections, the system needs to monitor a shared application. It should be able to tell where on the visual surface representations of private elements or elements that need verbosity adjustments are located, which visible dialogs, windows, menus and UI widgets are private, and if the application is in a private state.

This requires methods for obtaining information about the application’s GUI compo-

simply copying from the window’s device context will also copy parts of covering windows, but it is possible to use the new `PrintWindow` functionality to force a window to render itself onto a separate device context. See Tan et al. (2004) for more details.

nents, the underlying semantic objects, and their visual representations. The following query layers are used.

L1: OS windowing queries – Enumerating all windows belonging to a specific application (or process), detecting creation/destruction of such windows, visibility, titles and locations. Many of the widgets used in an application are themselves windows and can be accessed the same way. This layer also supports capturing of keyboard and mouse events. In the prototype we relied on using P/Invoke² to access Win32 dll calls that provide this information. Similar functionality is supported in other modern operating systems.

L2: Accessibility API – These are common APIs often targeted to sight-impaired users. They enable third-party tools, such as screen readers, to systematically expose information about UI elements in a running application. Exposed information contains element names, roles, text, visibility, state and on-screen location as well as events triggered by UI elements (as button clicks). It is also possible to walk the accessibility hierarchy to explore children items (e.g. dialog items within a dialog, as in Figure 4.3).

Previous work (such as by McGrenere (2002)) has already demonstrated how tracking Accessibility events (using a generic tracker tool from Microsoft) can be used to log a user's activity with an application.

We extended this information channel and successfully re-purposed these APIs as a resource to detect exposures and locations of elements that should be kept private or highlighted (e.g. UI widgets, menus and specific menu items, rendered HTML objects).

Accessibility APIs are now supported by many commercial tools, UI toolkits and operating systems.³ Our use of them can be further generalized.

L3: Application-specific API – Many commercial applications provide an API for integration and automation. These APIs can be used via COM and a scripting language (VisualBasic or C# scripts in Windows), JavaBeans, AppleScript and other frameworks. Within our system, we used these APIs in a simple manner to extract information about the application's state and to identify the visual representations of semantic objects. While writing some scripting code to work with the API is required, our experience when devel-

²Using P/Invoke is described in <http://msdn.microsoft.com/msdnmag/issues/03/07/NET/> (last checked August 17th, 2005).

³Supported by Microsoft Accessibility, Java APIs, OS X, wxWidgets and more. Some level of accessibility is now required by law (<http://www.section508.gov>, last checked August 8th, 2005) and will no doubt increase over time. Some standardization can be expected that will further enhance the range of platforms we can support.

oping the prototype shows that this is a focused effort with a limited amount of coding. Modern APIs already provide methods for locating document model objects on the window surface, or an Application object can usually be queried for its current state. Furthermore, coding occurs only once and can then be used in flexible ways.

L4: Extracting information from surface drawing operations – This is a technique introduced by Olsen et al. (1999) and is based on overriding the basic drawing routines so object locations can be extracted (see Section 3.5). Its requirements are quite problematic, especially for the commercial tools we worked with, therefore this technique was not used and it is not shown in Figure 4.1. However, it is still a possible semantic glue layer that could be used in some cases.

4.1.3 Plug-In Architecture

To create a generalizable framework, we chose to implement all of these queries using a plug-in architecture for our system. Each shared application has a middleware plug-in to our system ⁴ that provides the semantic glue and extends a generic base plug-in. Our architecture defines a Plug-in API (PAPI) with the services it expects from each middleware plug-in. Default actions are provided for each service in the base plug-in. Application-specific plug-ins can be added that override the default actions. The Monitor module can then use these plug-ins for parsing the visual surface and window set of an application and apply suitable filters.

Base Plug-In

A default *base plug-in* provides a set of general capabilities to track common UI entities. It serves as a toolbox for developing more specifically tailored plug-ins.

The base plug-in runs a background service that searches for dialog boxes, menus and other widgets (like dropdown boxes) of a shared application by tracking window creation events using L1 methods. It then uses additional L1 calls to extract their window class (type), title and location and the accessible object associated with the window⁵.

⁴The plug-ins we describe are added to our system, not to the applications. The plug-ins may have specific knowledge of the application and its API, but no access to the source code of the application is assumed or required.

⁵We used the `AccessibleObjectFromWindow()` and `AccessibleObjectFromPoint()` calls available in Windows, but there are equivalent methods in other Accessibility packages.

L2 calls are then used to extract accessible name, role, text, state, selection and location for the object and its child elements (e.g menu items or dialog fields) on the “accessibility tree”.

Hints on how to handle these items can then be obtained by testing this information against information specified in an application-specific plug-in without additional coding as will be described in the next section.

Handling UI components means making decisions about which components are to be replicated on the public screen, which components should be replaced with alternative representations or what image filters should be applied on the visual surface of a window where such elements are located. The base plug-in provides a set of generic image filters for blurring and highlighting that will be described later and the Director module can be instructed what components to replicate.

The base plug-in also uses the same accessibility information to provide generic augmentation capabilities, such as highlighting the active dialog field, highlighting menu selections, or telling the director to “gradually decay” the image of dialogs and menus on the public display after they are released by the presenter.

It is important to note that the base plug-in is application agnostic. It does not carry out any L3 queries, because these are application-specific. All application-specific knowledge is specified in extension plug-ins, or in cross-application policies that will be described later.

Application-specific Plug-Ins

An application-specific plug-in encapsulates the knowledge about a specific application and its monitoring, and supports a common API that the Monitor module can use. These encapsulate behavior that customizes or overrides the behavior of the default base plug-in. There are three main types of such methods (see also Table 4.1):

Privacy Hints – One set of methods of the Plug-In API (*PAPI*) provides privacy hints. When a new window or widget is detected by the base plug-in it will direct a call to the appropriate application plug-in with the Accessible object representing the widget and its properties as input.

The application-specific plug-in returns text or keyword-based descriptions and hints on the UI widget (indicating if it is private and should not be replicated, if it should be replaced with an iconic representation, or if it or any of its child elements need to be blurred). Most of

these services can be simply realized by checking the string based widget properties (window class, title, accessibility name, accessibility text, accessibility role) against a predefined classification table. In the prototype, which is a work-in-progress, widget classifications were hard-coded. However, one could represent these classifications in an XML table or simple scripted hints that do not require coding and process the information extracted by the base plug-in. For example, a set of rules that defines the “save as” dialog as private and replaces it with an iconic animation, and also makes sure that the “File” menu is blurred for a specific application, might be written in pseudo XML script as:

```
...
<Match wndClass="# EXCEL-DIALOG-CLASS #", title="save as">
<Classification status="private" hint="animation:filesave"/>
</Match>
<Match accessibilityRole="menu", accessibilityName="File">
<Classification status="private" hint="blur:pixelize"/>
</Match>
...
```

These *scripted hints* can allow a presenter to easily customize manipulations to the shared view without needing to write a separate plug-in for the application.

Other privacy hints can be associated with application-specific states (such as when switching to a private worksheet). These are extracted using application-specific L3 calls.

Visual surface parsing – Other PAPI methods are used to extract lists of regions of the visual surface containing private information, regions that need highlighting or specific sub-window areas to be displayed (instead of the full window). A plug-in translates these general PAPI queries into appropriate queries in one or more of the four layers. Usually this will translate to L3 scripting calls to locate specific document model objects on the visual surface (e.g. the selected paragraph, cells with certain attributes or cells that were changed) or applying L1 and L2 calls to locate sub-windows of the application on its window tree (e.g. locating toolbar windows and excluding them from the region to publish).

In the prototype we focused on writing a wide range of parsing capabilities in L3 and L2 for each one of the applications we worked with to test different approaches and ideas. Further research is required on how to customize these capabilities for specific presenter or

audience needs.

One possible approach that was partially taken in the prototype is to extend a plug-in A for application α that provides the basic queries, by creating a version A' (by inheritance) that overrides the relevant PAPI methods (Table 4.1). Another approach exposes the basic queries for meta-scripting (so, for example, a rule like `"cell[*,*].backcolor=red → blur"` will apply blur filters to every cell colored in red, given that checking cell properties and computing their on-screen locations are defined in the application-specific plug-in A).

Once scripts providing these basic capabilities are written, they can be driven by meta-scripts similar to the previous set of methods.

Application-specific resources and UI – These are methods that return application-specific animation clips, image filters and command descriptions stored in the plug-in to be presented to viewers.

When a plug-in classifies a widget or region it also gives back a hint on what filter, augmentation or animation to use. Some are cross-application (like a file open animation or basic blur filters provided by the base plug-in) and some are application-specific (a data import wizard animation for Excel provided by an Excel-specific plug-in). Both types of hints have been implemented in the prototype.

Many keyboard shortcut descriptions can be extracted automatically from the Accessibility information by doing a reverse mapping (the accessibility information for menu items usually contains the keyboard shortcut for them and the menu item name can serve as the index). In the prototype this approach was used in a manual manner (i.e extracting a subset of shortcuts from the accessibility information off-line and encoding the reverse mapping in the plug-in).

An application plug-in can also provide its own UI for tweaking the manipulations or even the scripted hints, as will be discussed in Section 4.4.3.

Plug-In Repository

The Monitor module directs its calls to a plug-in repository manager, which loads the appropriate application plug-in at run time (possibly even from remote servers). If no appropriate plug-in exists the default base plug-in will be used, offering some basic monitoring capabilities. (At its simplest, the base plug-in could query the presenter before displaying any menu or dialog box, and then apply “program by example” techniques according to the

Plug-In Interface	
Privacy Hints	
<code>GetApplicationState()</code>	Returns a set of keywords describing the application state and its privacy (e.g. "visibleWorksheet[params]",private). It uses the base Plug-In to obtain a description of active application windows and dialogs, and L3 calls to the application to obtain application-specific states.
<code>GetWidgetPrivacyHints()</code>	Gets as input the accessibility object describing the widget (dialog, window, tab etc.), as extracted by the base plug-in. Returns hints on the privacy level of the widget and its children.
<code>GetMenuPrivacyHints()</code>	Similar to <code>GetWidgetPrivacyHints</code> , but specific to menus.
Visual Surface Parsing	
<code>QueryPrivateRegions()</code>	Returns a list of regions in the frame buffer that contain private information (using L3 or L2 calls). Each region is returned with keywords describing it and hints on the suitable blurring effect.
<code>QueryHighlightRegions()</code>	Returns a list of regions in the frame buffer that need highlighting (e.g. active selection context, specifically selected document object). Each region is returned with keywords describing it and hints on the suitable highlight effect.
<code>QueryChanges()</code>	Returns a list of regions that map to document changes (e.g. text or cell changes) or global state changes to be reported as subtitles (e.g. switching sheets or document sections). Each change is returned with keywords describing it and hints on the suitable effect to use.
<code>QueryPublishedRectangle()</code>	Returns the subarea of the main application window to be "published" or replicated for viewers (based on the selected policy: follow a specific object, follow active selection context, exclude UI layers etc.
Application-Specific Visual Resources	
<code>GetVisualResourceFromToken()</code>	Given a hint token, returns a visual resource (such as an animation sequence, icon or effect operator).
<code>GetCommandDescription()</code>	Given a command (such as keyboard shortcut) that was captured by the base plug-in, returns an application-specific description of that command that can be used for reporting.
Plug-In Control	
<code>ProcessKeyboardInput()</code>	Handles keyboard commands that control the plug-in's functionality. For example, the Excel plug-in can be instructed to toggle between highlighting active table, active row, or active column.)
<code>ShowUI()</code>	Brings up a UI window (or a semi-transparent layer over the application's window) to control plug-in specific functionality. (Not implemented in the current prototype.)

Table 4.1: The Plug-in API (PAPI). There are four main categories of methods: (i) Privacy Hints on application windows, widgets and menus; (ii) Visual Surface Parsing to extract regions that contain private information or need highlighting on the application's visual surface; (iii) Visual Resources that access application-specific animations, icons or effects; and (iv) Plug-In UI and control to tune plug-in-specific functionality.

presenter's responses.)

A similar scheme is successfully used in popular web browser extensions⁶ that grab pre-written web-site-specific DHTML “user scripts” from online repositories. These control changes to the way the site looks and behaves in the web browser. These site-specific scripts (many are contributions of community members) behave much like the application-specific plug-ins we suggest. The source code for these scripts is provided freely so users can adapt them to their own needs.

Detailed Plug-In Examples

We next discuss two detailed examples of application-specific plug-in used in the prototype. These highlight the different ways semantic queries can be posed and generalized.

Plug-In Example I

Our Excel plug-in defines the keywords “open”, “save as” and “options” in a dialog title as private. When the base plug-in finds these in a dialog, a private state indication will be issued and the dialogs will not be exposed. The keyword combination `path="Format Cells/ Protection";role="tab"` will tell the base plug-in that the tab widget entitled “Protection” in the “Format Cells” dialog is private (Figure 4.3) and its entire contents should be blurred. The path regular expression: `"File/(^.*\.xls$");role="menu item"` orders all items in the “File” menu that match a file pattern to be blurred (i.e. recent MS Excel files). Some of these rules were hard-coded in the prototype.

To get regions for blurring or highlighting, Excel-specific API calls (L3) are used to locate cell ranges marked in a specific background color, the selected cell range and its surrounding table, or changed cells. The on-screen bounding box is then computed.

Specifically, Excel.Range objects provide color, bounding box and value properties that are simple to use and the Excel.Application object has methods to get selection and changed ranges. If the selection is in a private cell range, additional L1 and L2 API calls are used to locate the formula edit box or locate the sub-window frame containing the document (these can be identified by their window class or name and position in the window tree). Once a bounding box is obtained the base plug-in provides generic blur and highlighting filters that work on any image buffer.

⁶A prominent extension is for Mozilla: <http://greasemonkey.mozdev.org/> (last checked August 7th, 2005).

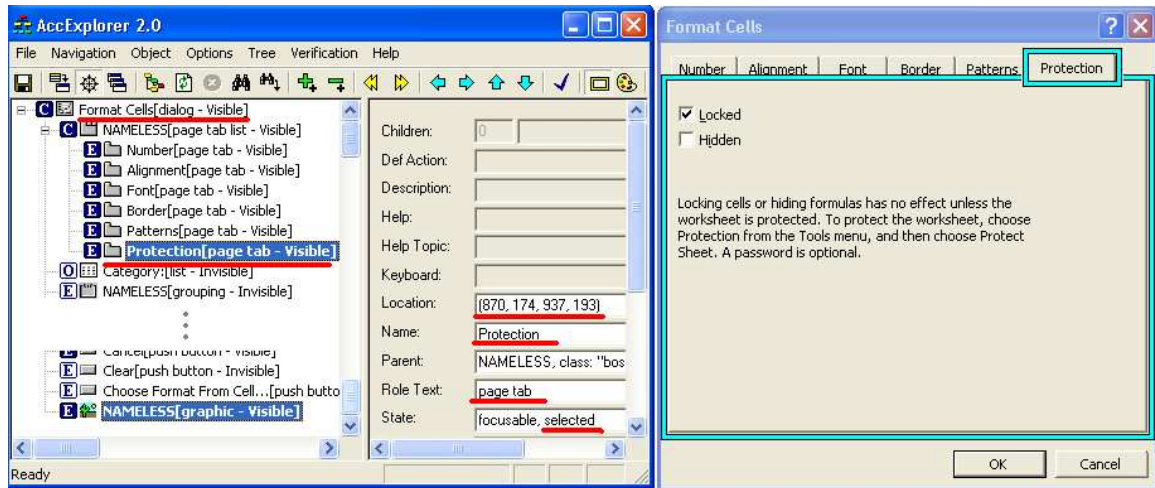


Figure 4.3: The Accessibility tree of an Excel dialog can be used to locate private items (such as the Protection tab) and their location. The Accessibility tree view in this picture was generated using the Accessibility Explorer tool (part of the Microsoft Active Accessibility 2.0 SDK).

In Excel, the Application object provides attributes for determining the active worksheet and the visibility of comments. Knowing this information, it is possible to enter a private state in Excel when particular worksheets are being viewed or when certain types of comments are present.

All of the above were packaged into a straight-forward C# script in the Excel plug-in. No changes were made to Excel, although knowledge of Excel's object model and API was necessary. Similar techniques were used for MS Word (the Range object for Word relates to a text range, not a cell range).

Plug-In Example II

Our web browser plug-in defines the keyword “favorites” in any menu as private (Figure 4.4). It also defines keywords matching the navigation history dropdown and auto-complete box (their window class, accessibility name and role) as private. Thus when any of these is opened and identified by the base plug-in, a private state will be issued and they will not be echoed on the public screen.

We targeted our plug-in to Internet-Explorer, but a similar plug-in for a different browser only needs to replace the keywords with the appropriate names (e.g. “bookmarks” instead of “favorites” for the Firefox browser). Alternatively a generic web browser plug-in can use a pattern like: `"favorites | bookmarks | ...";role="menu"`.

In a similar manner, L2 calls can expose the accessibility object associated with any HTML item by its title attribute (adding this attribute in the HTML code is now encouraged to assure that web sites are accessible), so any browser plug-in can easily locate a specific HTML form field based on the title value and then blur it or highlight it if appropriate.

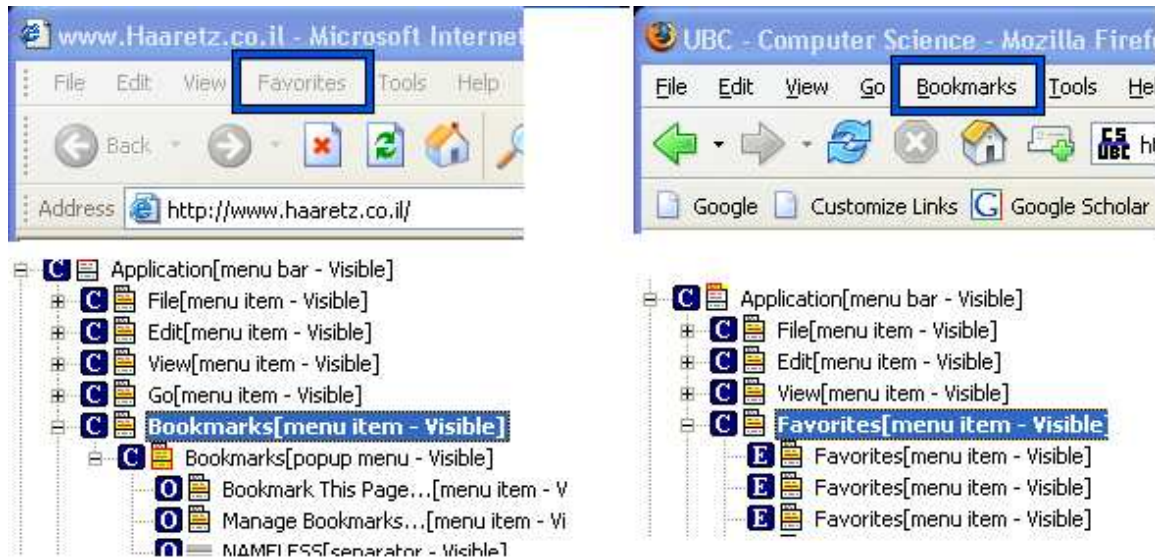


Figure 4.4: The Accessibility tree for the menu bar in Internet Explorer (left) and Firefox (right). By specifying the keywords “Favorites” or “Bookmarks” as private a browser plug-in can prevent these menus from showing on the public screen.

4.1.4 Policies and rules

The Director module handles the published representation to be rendered on the public display. It uses the Monitor module to track the application and extract descriptions of its state and visible elements. It then applies policies that determine how to manipulate the visuals.

When instantiating a policy, a tuple comprising the application, the state or element, and the viewer’s role is the input. The output is a rule that determines the manipulations that will be applied to the published visual representations.

Privacy classification

We must determine how private elements, states or elements that need verbosity control can be extracted, assuming applications being shared do not know about privacy. There are two complementary approaches to consider.

The first approach (taken in the initial prototype and partially discussed in previous sections) is letting the presenter mark these elements explicitly.

When working with a document in an editor we can readily support what we call a “*Magic Marker*” that maps a visual property of an object to a privacy state (most editors have a notion of object style properties). For example, a presenter can mark a document object as private by coloring it with a specific color, using the native application tools (e.g. a background color for cells in Excel or a highlight color for paragraphs in Word as shown in Figure 4.5). When writing in this color the PAPI calls translate to simple L3 scripts that will recognize these objects as private and extract their on-screen locations.

A policy that regulates blurring for marked objects will create the effect of a marker that cannot be seen by viewers, while the presenter can interact normally (as opposed to using black on black writing or using Excel’s column and row hiding features that will prevent the presenter from viewing and interacting with the information). This mode provides visual feedback and awareness on what the audience cannot see, as called for by Shoemaker and Inkpen (2001).

Other means for coding attributes can also be used (like adding a “Private” prefix to a worksheet’s name or to a comment’s text to mark their privacy). Another option is to use the application’s built-in selection mechanism, so for example the paragraph containing the insertion point can be extracted by querying the application and then it can be rendered differently.

A second approach is to use an automated rule-guided search for privacy *leaks*. We already described how the base plug-in can be augmented with application-specific verbose mappings. These can be extended to conduct online cross-application rule-guided searches for private information in any menu, dialog or document rather than relying on pre-computed classifications. By searching the text and context of UI widgets it is possible to identify error messages, dialog field names related to security or network settings and classify personal information appearing within the shared document.

We have experimented with searching for private text in a spreadsheet or document (phone numbers, names, etc.) and automatically blurring them. Another interesting domain is that of web pages, where we partially implemented a search of the HTML code for private UI widgets and content elements (e.g. examining all form field names and blurring ones that might contain private information, such as userids, credit card numbers, etc.).

Sample Rules

We believe that a combination of these approaches is required for adequate privacy protection. Together with the visual manipulations (described later), this allows a flexible range of rules or policies. We list a few examples of the types of rules that we have considered in our prototype.

- “Do not expose any dialog related to files or the network in any application to any public viewer.”
- “Blur any document element in any application marked in pink to group A members.”
- “If there is any viewer from group B, do not expose my “favorites” in a public view or any web page not coming from company servers.”
- “Ask me before exposing any window from application X on the public display, unless I authorize it beforehand.”

Also useful is an opt-in policy of “blur everything unless specifically marked by me” as opposed to the opt-out versions we used in the prototype. Opt-in policies can better protect against unexpected exposures of information, but require more work and attention from the presenter.

Specifying Rules and Policies

In the prototype we have concentrated on the system architecture and a collection of manipulations (discussed in the next section) that illustrate the value of having role-based policies to control views.

Our goal has not been to develop a robust mechanism for describing policies. However, a prominent future direction is looking into adapting schemes that allow policies, roles and rules to govern access control in collaborative sessions (“login space”) to regulate privacy (“display space”).

Schemes similar to the ones proposed by Edwards (1996) may be used to specify static or dynamic policies in a flexible policy specification language. The schema presented there already allows rich ways to describe roles and policies some are evaluated based on the outcome of scripts. The missing link is the rules that identify the visible UI elements and

semantic objects and what privacy or augmentations filters to apply. The simple scripting and query examples shown in this section could serve as a basis for such a language. This is a fruitful area for future research.

4.2 Manipulating the Visual Representation

The Director component takes in the “raw” captured frame buffers grabbed from the application windows and applies one or more of the following manipulations based on the policy and rules that are in effect.

4.2.1 Blurring

When private elements are visible, the challenge is guaranteeing that viewers cannot see them while allowing the presenter to work freely. The PAPI can extract the locations of such elements on the visual surface at any time (with attributes and hints, such as the suggested blur effect to use). In some cases a private information unit may appear in several places (e.g. the contents of a selected private spreadsheet cell will also appear in the formula bar). This demonstrates why tighter integration with application semantics is crucial for ensuring privacy.

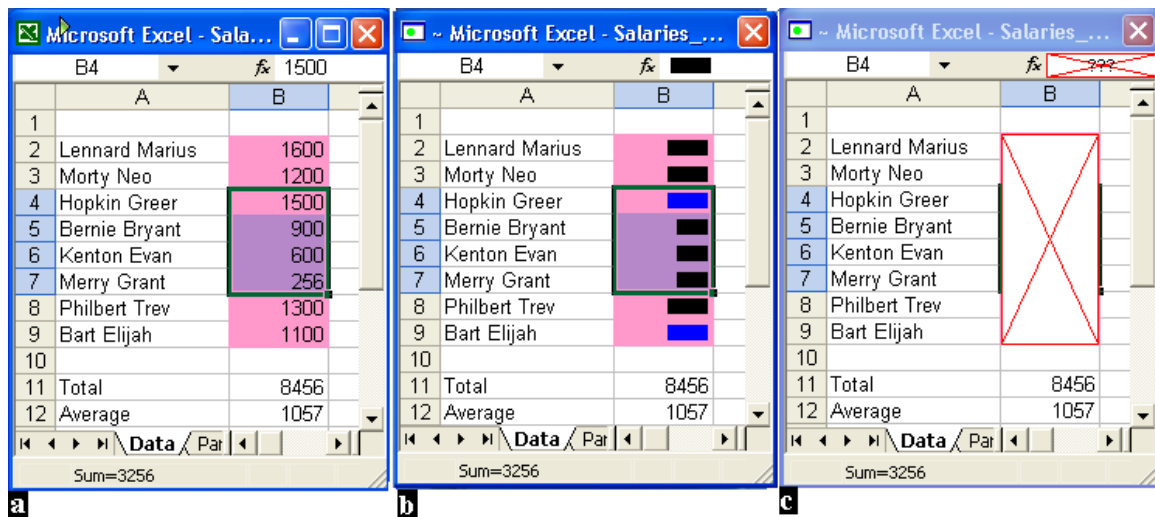


Figure 4.5: (a) The presenter’s view of a spreadsheet, (b) Greeking cells marked in pink, exposing selection and style, and (c) fully concealing a cell range.

The Director can apply several image blurring operators on extracted private zones (Figures 4.5 and 4.6). Because blurring occurs at the frame buffer level it can be applied

regardless of what the underlying element is (UI control, text, image etc.) or how the bitmap was originally drawn (therefore all filters used in the prototype are provided by the base plug-in). Different filters offer different visual affordances, balancing between the presenter’s privacy and the audience’s awareness.

- **Draw over** – Invoked for full privacy, with no awareness of the presenter’s interactions in the blurred part.
- **Greekify** – Creates a “Greeked text” effect by searching text line boundaries on the image and replacing them with filled rectangles in the dominant color. The filter implementation resembles the techniques used by Olsen, Taufer, and Fails (2004) for finding text paragraph boundaries from a hand-sketched annotation. The major difference is that our system uses automated visual-surface parsing to direct the filter as opposed to reliance on manually generated annotation marks. This filter is useful for exposing structure, style and some notion of the presenter’s interactions, such as selection, without exposing sensitive text.
- **Pixelize** – This is a general purpose filter, mostly useful for image-based content. It provides awareness cues for viewers, but may be insufficient for full privacy.

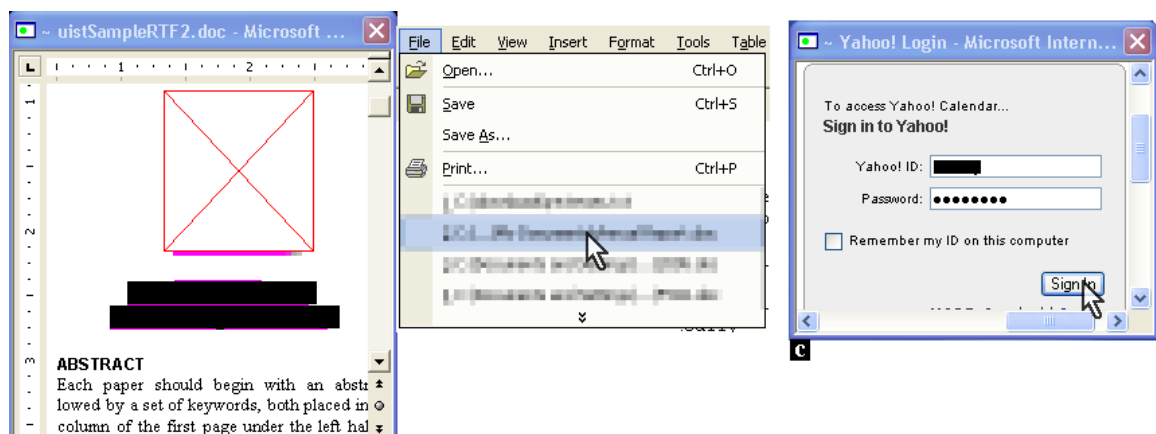


Figure 4.6: (a) Word document with blurred paragraphs and image; (b) recent files menu items blurred by pixelization; and (c) a login web page where the userid field was detected and Greeked.

4.2.2 Saliency and highlighting

The system supports a highlighting mechanism that is independent of the shared application's own selection and highlighting tools. Highlighting is used to draw viewers attention to important content or changes (also serving as visual deixis), many of which are not identical to the current selection. Earlier work by Olsen et al. (1998) looked at allowing different agents, software and human, to point into the visual space of an application and some of our highlighting shares the same visual effect.

In Figure 4.13a the presenter is interacting with the tools palette, but wants to keep viewers focused on a specific paragraph. The PAPI provides a method through which the shared application can be queried for regions to highlight, supporting policy-guided highlighting.

Highlighting active selection context

We found it useful to highlight the context for the active selection as changes are made. The context is application-dependent (the paragraph, sentence or section containing the insertion point in Word, the table surrounding the selected cells or dependent cells in Excel, or the active dialog field in any application).

The highlighting effect is application-independent and works on the image buffer by placing a semi-transparent colored mask on top of non-highlight areas. Detaching the highlighted object from selection and instead highlighting a specific object, while working with other objects, is also useful and can be done by caching the previous highlighting bounds or by caching a pointer to the previously selected object within the plug-in.

Highlighting changes

Another mode of highlighting makes changes more salient to viewers (mostly to indirect changes in parts of the visual surface far from the presenter's interaction). A considerate presenter would point out these changes to an audience and perhaps even mark them on the screen. To assist the presenter, the semantic glue layer can extract such linkages and provide automatically generated highlighting. Figure 4.13c shows a hand-drawn style of circling for changed cells that exposes changes in blurred data while preserving privacy. When working with minimized screens it is also useful to highlight and intensify changes

of state that are otherwise hard to detect, such as when the presenter switches between worksheets, sections or documents (Figure 4.13e). This can be accomplished by reporting the new states as subtitles.

Other manipulations that can affect salience and attention are the magnification of relevant regions of the visual surface, re-rendering of textual elements in a bigger font (many of these underlying texts can be extracted through the semantic glue layer) or “shaking” windows or parts of the windows that have changed.

4.2.3 Spatial manipulations

One set of manipulations allows the presenter to share only a partial view of an application’s window. This is useful for reducing screen space use and clutter, and in addressing privacy. The PAPI provides a method through which the window part to be shared (“published rectangle”) can be accessed. Computing this window part can take into account several policies.

- **Excluding the UI** – Remove UI layers such as toolbars and embedded windows that take a substantial amount of screen space (Figure 4.7b). L1 and L2 calls can be used to search and prune the application window tree.
- **Active context** – Share only the active context, based on the presenter’s selection using techniques described in the previous section.
- **Sharing a specific element** – Share only a specific semantic object (paragraph or table) or UI element chosen by the presenter.

The semantic layer guarantees that the window part computed will adhere to the stated policy and will take into account changes to the window’s dimension, scrolling or UI changes, unlike the manual definitions presented in WinCuts (Tan et al. 2004) or the even less helpful fixing of a portion of the screen to be shared.

Another set of manipulations uses affine transformations. Rotating windows is useful for single tabletop display sessions, where viewer orientation should be part of the policy. Automatically scaling down the size of dialog boxes, palettes and other secondary windows (identified by the PAPI), together with a careful placement of these next to the full sized main window can assist in reducing clutter and support privacy (Figure 4.7c). Combining

several spatial transformations together can be quite powerful. For example, it is possible to publish only the selected paragraph context, flipped vertically so a viewer on the opposite side of a tabletop display can follow the discussion without requiring replication of the full document window (Figure 4.7a).

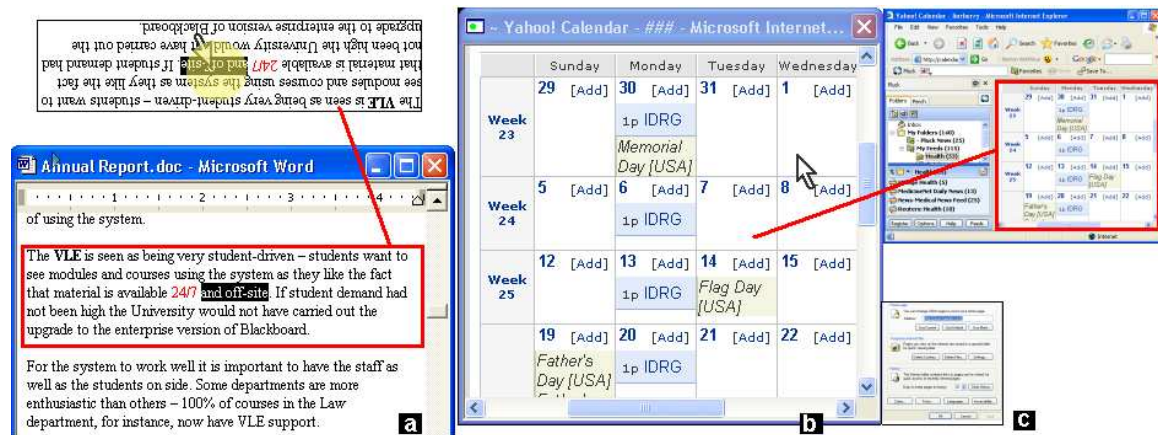


Figure 4.7: Spatial manipulations: (a) publishing only active context (paragraph containing insertion point) + vertical flip; (b) auto-exclusion of toolbars, menus and embedded frames from an Explorer window, exposing only the sub-window with the HTML page; and (c) automatically downsizing an “options” dialog.

4.2.4 Temporal manipulations

There are situations when it is more reasonable to define the entire state of the application as private, rather than extensively applying blurring transformations (e.g. when the presenter interacts with a private worksheet, uses the file open dialog, or works with a wizard).

The semantic glue layer can query the application state. If it matches the privacy policy, the system can trim the interaction timeline by not sending updates to the viewer’s display until the presenter exits the private state. To keep some level of awareness for the viewer, the system can display an animated iconic representation summarizing the state. For example, an open file icon can appear instead of exposing the dialog itself (Figure 4.8a) to prepare a viewer for a document change, or an icon indicating interactions on a private worksheet can give the viewer hints about what the presenter is doing.

In other situations, it is better to not provide any indication at all, maintaining complete privacy, as when the presenter interacts with an auto-complete text box (Figure 4.8c), an error dialog, a private comment, or commits syntax errors when taking notes in public.

Some of these states are quite unpredictable by the presenter (e.g. an error dialog popping up or auto-complete suggestions), so automatic detection of these is crucial.

Another set of timeline manipulations can be applied to the pace at which certain operations (as extracted by PAPI) are played on the public display or by letting viewers roll back recorded interactions (for example, using the semantic glue to tag recorded interactions for quick roll-back). One example we implemented involves menu selections, discussed in the next section.

4.2.5 Handling Menus

Menus are fundamental interaction components that are often problematic in a generalized presentation scenario. They can create a lot of clutter (being arbitrarily long, regardless of the size of their parent window), and they often bundle private information (recent files, bookmarks). More importantly, menus are becoming highly tuned for the active user and less for a passive viewer (adaptive menus with personalized order and gestural menus that do not show on the screen). As described previously, the base plug-in obtains the relevant attributes (name, items, locations, selection) from any menu through L2 calls and can mandate blurring on specific items or prevent a menu from showing on the public display (these are highly generalizable techniques).

A critical moment is when the presenter makes a selection on a menu. From her point of view there is no need for the menu anymore and it is taken away by the application. Our system, however, can capture the selection event and pause the timeline so that the menu lingers on the public screen for an extended “decay” period suitable for passive viewers (possibly with animated highlighting of the selected menu item). It is also possible to identify the creation of a pop-up context menu (which might result from a right mouse click on a link in a browser), and smoothly move its clone to a neutral placement that does not block the operation’s context (Figure 4.9a). We can use these techniques to stall the progress of a dialog on the public screen after the presenter closes it, allowing viewers to fully understand the interaction.

Watching interactions with menus is not the best way to convey operations to viewers. In many cases, replacing a menu selection with different feedback, such as specifying the selection in a semi-transparent subtitle, is a better technique (Figure 4.9b). Consider a presenter who scrolls through a menu until finding a specific item. It is hard for a viewer

to tell if the menu was closed because a selection was made or the menu was released with the ESC key without making a selection. Our subtitle scheme reports only when a selection is actually made. This scheme still works even if the presenter uses keyboard shortcuts or gestural menus because the semantic glue translates these back to a menu item description for display, so all equivalent operations appear the same to viewers.

4.2.6 Mouse Cursor manipulations

We are able to provide viewers with a different representation for the mouse cursor to better suit their needs, while the presenter can continue working with a normal cursor.

Some of these alternative representations are shown in Figure 4.10. They assist viewers in visually tracking the cursor and knowing the presenter's focus of interaction. Alternative cursors can support augmented or relaxed verbosity. They can increase verbosity by visually encoding low-level interactions (such as mouse button clicks or keyboard presses) or reduce it by displaying only the state indicators discussed previously. Because passive viewers often focus their attention on the moving cursor, encoding indicators on the cursor (as in Figure 4.10d) will grab their attention.

Finally, it may be desired to conceal the mouse cursor altogether when the presenter interacts with components that are private, so mouse positions do not disclose interactions.

4.3 Access Control Extension for Input

Our focus so far has been strictly on output. We also want to support limited forms of shared input, with an emphasis on preserving privacy.

In augmented collaboration environments like the iRoom that uses PointRight (Johanson et al. 2002) or similar solutions (Rekimoto and Saitoh 1999; Booth et al. 2002), or in remote desktop solutions such as VNC, a presenter can grant a viewer full control over keyboard and mouse input to her machine (a coarse form of floor control). For example, when requiring help (as in Scenario 2.3), when delegating a task, or when letting a viewer present something from his machine.

This is highly undesirable, because from the presenter's operating system's point of view, keyboard and mouse input are still coming from the logged in user (the presenter). Therefore, a viewer who takes control could make changes to non-shared applications or

make unwanted changes to the application being shared.

We integrated the basic functionality of PointRight into our system (for example, allowing a viewer to control the shared application from her PDA as in Figure 2.3), but we are able to use the semantic layer to identify locations of widgets, menus and controls that should not be accessed. Thus when a viewer sends a mouse click event on such a control, the system will not pass this event to the OS. Similar treatment can be applied to keyboard events (this requires more queries on the application). This scheme allows finer grained access control policies to be applied to applications that were not designed for it. Additional work is required to make this technique more robust.

4.4 Feedback and Control

Solutions that address privacy differ “by the degree to which subjects have feedback about and control over the disclosure process” (Lederer et al. 2003). Our system is aimed at providing semi-automated control of privacy based on rules defined by the presenter. There are two problems that arise:

1. Privacy and augmentation manipulations are done on the public view of an application, so a presenter may forget what viewers can see or may not be aware that some elements are kept private and do not get copied to the public view.
2. In some cases the privacy attribute assigned to a GUI or document item based on the rules can be incorrect or may not suit changes in the circumstances. For example, the presenter may have specified in a rule that any context menu should remain private, however, at some point he would like to explicitly show an operation involving this menu.

The two problems demonstrate the need for feedback and control mechanisms in the presentation time.

4.4.1 Radar View

One approach taken in the prototype uses a radar view window on the presenter’s screen to provide constant feedback on what the audience can see (Figure 4.11). In co-located scenarios the presenter may still be able to look at the public screen (although it may

actually be behind the presenter). In distributed scenarios knowing what viewers see is crucial for successful collaboration. Radar views are an efficient mechanism for maintaining mutual awareness in groupwork environments (Gutwin and Greenberg 1998b).

A presenter can also use the radar view to control the public copy of the application through window miniatures (for example, to move these windows on the public screen). Early VR work has already used the concept of a world-in-miniature (Stoakley, Conway, and Pausch 1995) for navigation and view control. Similar techniques can be used for shared viewing.

4.4.2 Changing Privacy Classification

Other feedback cues that we experimented with in the prototype can serve as in-situ privacy controls. Figure 4.12 left shows a balloon window that shows up when the presenter starts interacting with a dialog of a shared application.

This balloon provides an alert to the presenter that this specific window is either visible or invisible to the audience (based on the privacy policy and rules). It is possible to add a button to this balloon so the presenter can override the privacy classification.

Another example (Figure 4.12, right) uses a more subtle privacy cue. A mini-icon window is attached to widgets that are not visible on the public view (in this case a menu) and provides privacy feedback to the presenter. When the presenter's mouse lingers over this mini-icon it will expose the menu on the public screen. Using this scheme solves a problem typical to menus - if the presenter had to click on the screen the menu would have disappeared, because this is how menus are programmed to behave.

4.4.3 Plug-In UI

Plug-Ins for specific applications may require controls that are application-specific for fine tuning visual effects. For example, the Excel plug-in provides highlighting of the active selection context, by marking the boundaries of the table containing the cell selection or the formula bar if it is being used to edit a formula. However, in some cases it is desirable to highlight the active row or column or fix the selection on a specific range of cells.

One possible solution relies on keyboard shortcuts. A special keyboard shortcut (Ctrl-Ctrl in the prototype) will transfer keyboard focus to the clone window, so following keyboard events can be used as commands to alter the plug-ins behavior (via the ProcessKey-

boardInput() method, Table 4.1).

For example, pressing the right arrow will toggle between table, row and column highlighting for Excel. A Word plug-in may use other keys to toggle sentence, paragraph or section highlighting.

Another option is for the plug-in to provide a set of UI controls that can modify its behavior. Thus when the special keyboard shortcut is detected the plug-ins ShowUI() method will be called and the control window will be displayed on presenter's display, either as a separate window or as a semi-transparent layer on the shared application window that was found to be effective in the Notification Collage (Greenberg and Rounding 2001). This functionality was not implemented in the prototype.

A possible extension to the system relates to editing the scripted hints that govern a plug-in's behavior (discussed in Section 4.1.3). Tools like Microsoft's Accessibility Explorer (see Figure 4.3 and Figure 4.4) already provide a general interface for selecting application UI widgets. A relatively simple extension can allow a presenter to visually select a specific widget and then provide or alter the privacy specification for it. These interactions could be translated to the underlying scripts.

4.4.4 Audience input

Apart from letting an audience member control the shared application (as discussed in Section 4.3), input from viewers can be used to adjust their view properties, control the pace of the presentation, or replay specific parts on personal displays. None of these functionalities was implemented in the prototype, but we discuss each of them here because they are easy extensions to what has been implemented.

Replaying interactions

Another functionality that can greatly assist passive viewers is the ability to replay recorded interactions (either off-line or while the presenter is still talking). The desire for such functionality was also expressed by the participants of the user study testing our system as will be discussed in Chapter 5.

The semantic glue can be used to make replaying more efficient by providing rich indexing of the recordings and allowing different modes of replaying instead of the time-consuming serial access:

- “Take me there” – Instead of “blind rewinding” backward in time, the meta-data collected by the semantic glue can be queried. Thus, a viewer could simply look for a specific interaction. Such an interaction might be an application-specific event (e.g. the point where the presenter changed a specific paragraph or switched to a specific worksheet), the beginning of a specific dialog interaction, or the point where a certain dialog field or a certain menu item was used. Such indexing is also useful for simply taking the viewer to the beginning of the most recent interaction sequence, which provides an answer to Baecker’s (1990) “how did I get here?” question (see Section 3.10).
- “Executive Summary” – Rather than replaying recorded interactions frame-by-frame it is possible to use the indexing to skip parts where no significant changes were made. For example, rather than watching the presenter type in the contents of a field, rolling the recordings to the point where she moved to the next field or to just before releasing the dialog may be enough. Another example would be playing recordings and fast-forwarding on the parts that did not involve changes to a specific paragraph.

Controlling pace

Viewers should be able to send feedback to the presenter throughout the presentation to control the pace of the presentation. While this could be done using a conventional communication channel (verbally, phone or chat) the process is quite awkward and a busy presenter might ignore such requests.

Our system is not intended to replace existing communication channels, but possible extensions to it could shift the control balance towards viewers. For example, we intend to explore ways of using the semantic glue to detect interactions for which viewers may want extended viewing time (such as when the presenter releases a dialog). In such cases viewers can get a short-timed popup control that allows them to freeze the dialog image on-screen and examine the changes made, or even replay the entire interaction. A companion extension on the presenter’s end could trigger a blocking mechanism to prevent the presenter from performing other actions until viewers release the lock.

4.5 Limitations

Experimenting with the prototype for the system showed that the proposed techniques, especially those relying on Accessibility APIs (L2), can work well with off-the-shelf applications. The system does indeed provide a useful and simple solution for sharing a view of an application with relaxed WYSIWIS and minimal demands on the shared application. However, some limitations of our approach were also evident.

4.5.1 Identifying private information

Tracking and handling automatically all cases where private information can leak is not always trivial. This is even true for collaboration-aware tools, but it is even more important for the collaboration-unaware tools our system is designed to augment. The level of privacy protection that is easy to provide is mostly bounded by the capabilities of the application's API and the quality and integrity of the Accessibility information. These are constantly being improved in new versions of software packages and UI toolkits, but may still be insufficient.

In other cases it is possible to rely on tagging and marking of private data, but this will require extra effort from the presenter. Furthermore, because we work with applications that are collaboration-unaware, providing a tagging mechanism for the presenter is not always possible (we used color or style attributes available in editors, but other applications may not have equivalents). In addition, if tags are not grounded in the document model they may become invalid under certain operations (for example, copying private data to a different location or using find and replace).

Still, in simple presentation scenarios the techniques we used may be enough and might be preferred over more complex solutions (look back at the privacy risk management criteria in Section 2.4.3). It is reasonable to believe that the more crucial privacy is for a presenter the more willing she is going to be to mark and tag these items (changing the document is also possible, but may prevent the presenter from viewing or manipulating the private data that is required for the task).

Another possible path is relying on UI customization and document model extension APIs that are now part of many off-the-shelf software packages.

4.5.2 Working on the image buffer

Private information is handled only on the visual surface level and not in the underlying document model level. We rely on image buffer and window set manipulations that do not require knowing how the visible information was generated and rendered. This provides an important advantage – manipulations and filters can be applied transparently and across applications.

However, using blurring filters or eliminating private information on the public display can still expose some private properties of the data. For example, the Greeking effect in Figure 4.5b exposes orders of magnitude for the blurred salaries (a more careful choice of the blur filter could have solved the problem in this case).

In addition, concealing information is itself an indication of the nature of the data. In other cases viewers may deduce the properties of blurred data from other visible information or operations (e.g. sorting a worksheet by salary will reveal who has the top salary).

These problems are common to other possible privacy solutions. Even when using a replication-based solution one still has to be careful what information is synchronized and how, so privacy is not violated.

4.5.3 Performance

There is an innate limitation on the public display update rate, because we copy image buffers. In theory performance should match that of VNC (assuming some variation of the RFB protocol is applied instead of the inefficient timer-based copying used in the prototype). In practice, the overhead of adding semantic glue queries and image buffer filters can slow down the update rate.

Our experience showed that the slow down was sometimes noticeable yet reasonable. The image filters and window set manipulations are quite efficient. The more expensive operations tend to be the ones involving application API calls.

It is clear that some of the APIs we use were not designed with performance in mind or even to work in all application states. API calls varied in execution time between application states and even resulted in errors in some cases.

Another problem with our approach is synchronizing image buffer operations (grabbing and copying) and the semantic glue parsing. This problem is quite similar to the one faced

by VNC, where unsynchronized updates or glitches occur from time to time (Lok, Feiner, Chiong, and Hirsch 2002).

The problem lies in the fact that grabbing the application’s image buffer is done out-of-process (with respect to the shared application). Thus, if capturing a paint event and analyzing it (as VNC does) or processing a semantic glue query take a non infinitesimal time t_0 , when proceeding to grab the changed parts of the image buffer, the analysis will be missing possible changes that occurred in $t_1 < t_0$. VNC hides even more complexity because some paint events are actually captured before the application has done the repainting.

Some of the heuristics used in VNC, such as sending custom synchronization messages to the shared application or using timer-based polling to update the shared view only in “idle” times⁷ can also be applied in our system.

In practice update glitches occur mostly when large parts of the visual surface are abruptly changed, such as in scroll operations or if the semantic glue query fails or takes a significant amount of time. They don’t occur in the typical case.

4.5.4 Feedback for the presenter

In the simple implementation of the system prototype only a special view for the audience is generated. The presenter continues to interact with the original application.

Therefore, the presenter may be missing critical cues on what the audience can see or what is highlighted on their view. The limitation emanates from the fact that manipulations occur at the image buffer level and cannot be directly applied to the original copy of the application.

In Section 4.4 some heuristics to address this problem were presented, but may be insufficient. Another possible solution is to treat the presenter as a special case of viewer. Instead of running the shared application on his computer, it can be hosted on a third machine (as already happens in some collaboration scenarios), or it could render into off-screen memory so it can be manipulated and re-displayed on-screen. Thus, the presenter’s copy of the application can be treated in the same way the audience’s is, but with a different set of privacy and augmentation filters.

⁷see a technical discussion on different VNC Hooks in <http://grox.net/doc/apps/vnc/winvnc.html> (last checked August 17th, 2005).

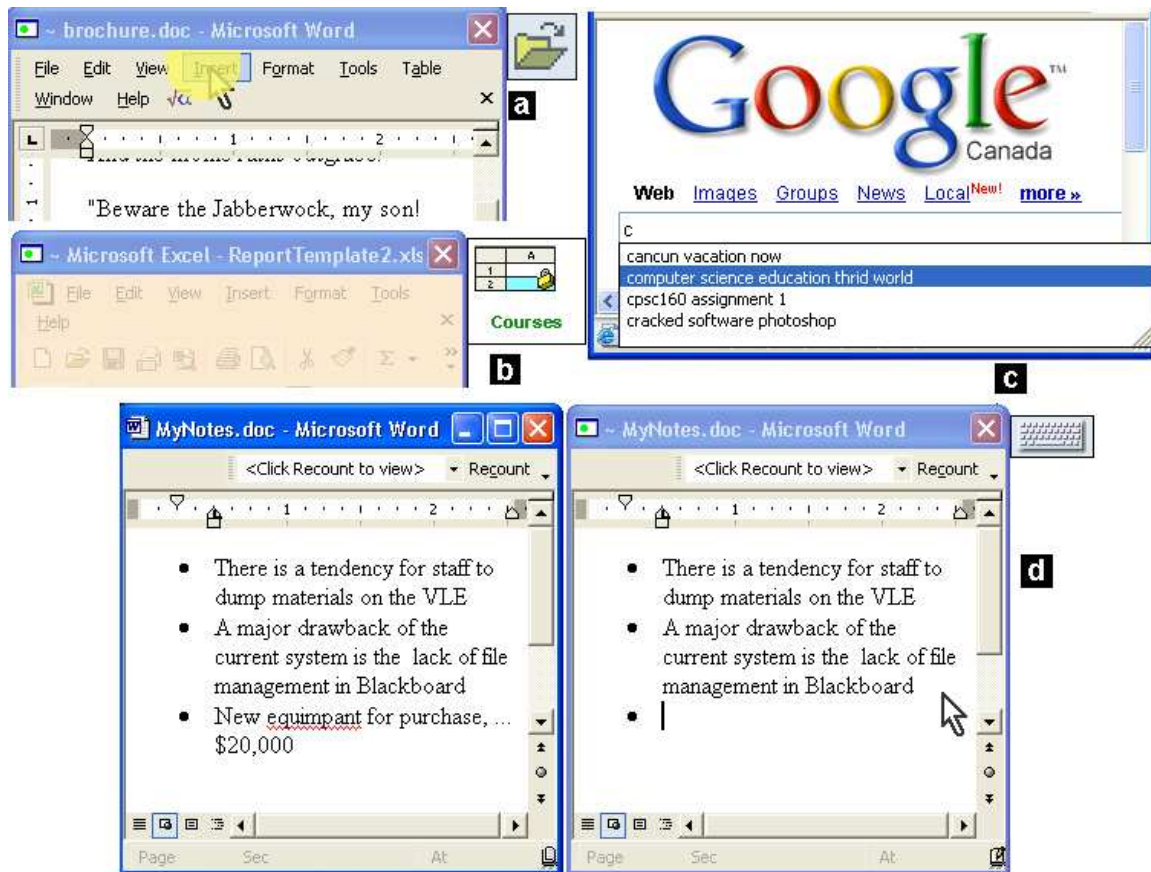


Figure 4.8: (a) The file open dialog is dynamically replaced with an iconic representation. (b) A private worksheet that has become active (marked by having “(Private)” as part of its name) is detected and replaced with an iconic indicator. (c) An auto-complete text box in a browser (in this case containing previous search items) is detected and not exposed to the audience. (c) The presenter is taking notes (d, left), so the public view (d, right) shows that the presenter is writing (a keyboard animation icon appears) but the view will be updated only when moving to the next bullet (pressing enter), so the presenter can fix syntax errors and expose notes to the audience only when ready.

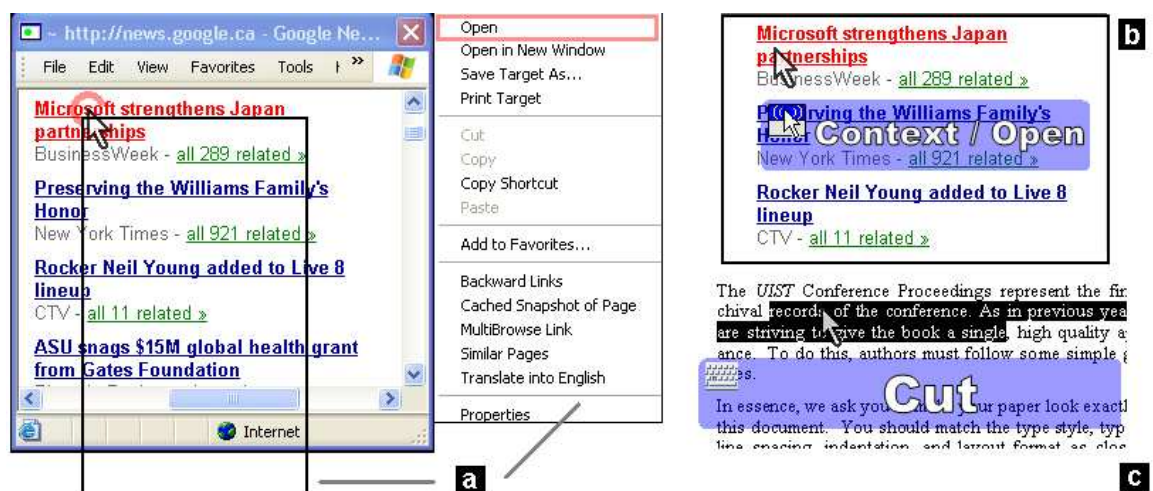


Figure 4.9: Manipulating menus: (a) moving a context menu to the side + highlighting selection after release, (b) replacing a menu selection with a subtitle below the mouse cursor, and (c) reporting a keyboard shortcut command.

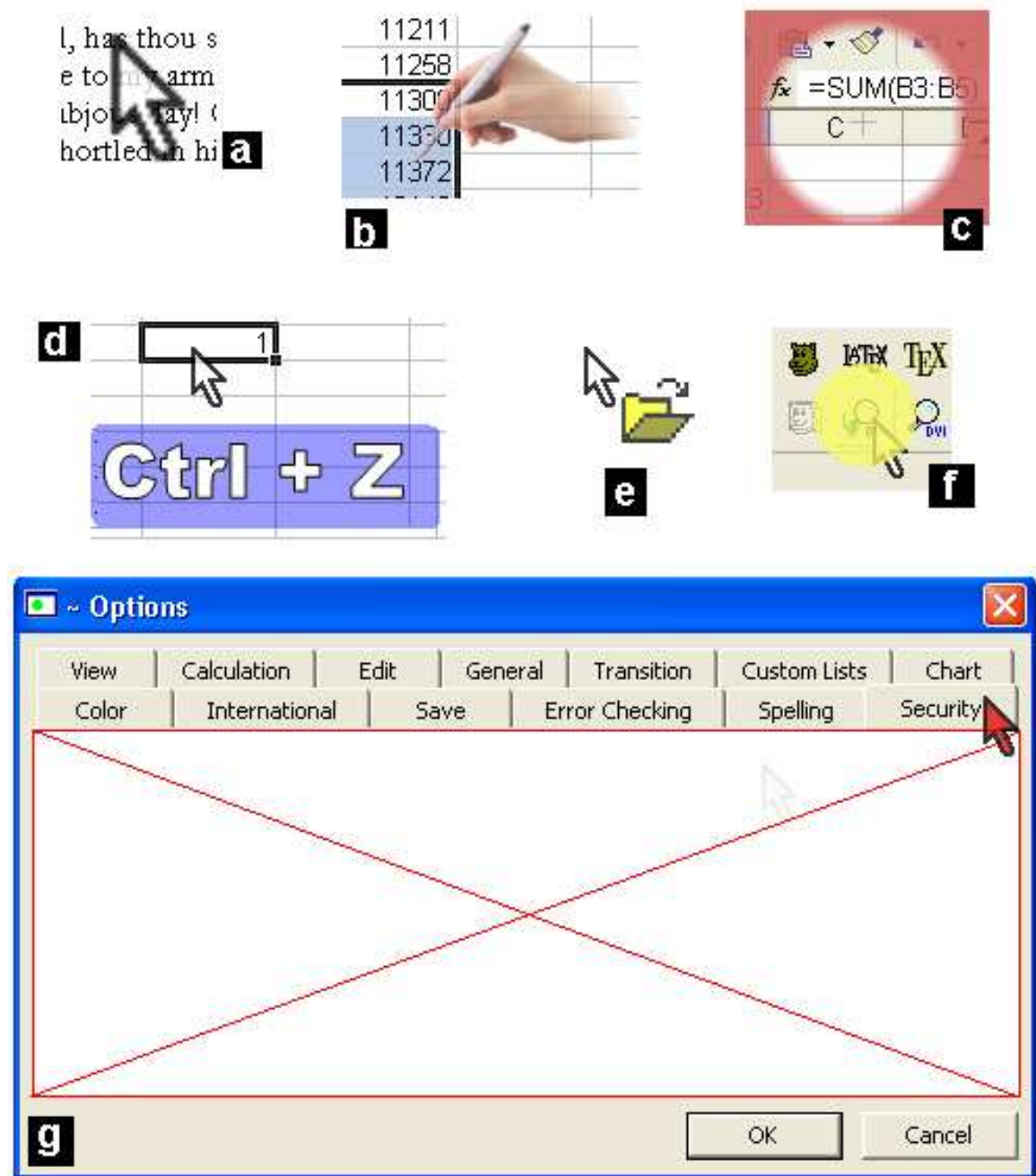


Figure 4.10: (a, b) changing cursor size, shape and transparency, (c) spotlight focus+context cursor, (d and f) visual feedback on the presenter’s mouse clicks and keyboard presses, (e) a private state indicator embedded on the cursor (file open) and (g) the cursor is frozen when going into the privacy-protected security tab, so its current location – shown here as a “ghost” – is not visible to the audience. The original cursor is shown in red indicating it has access control, i.e. if a viewer is given remote control of the mouse his clicks on the security tab will not go through (see Section 4.3).

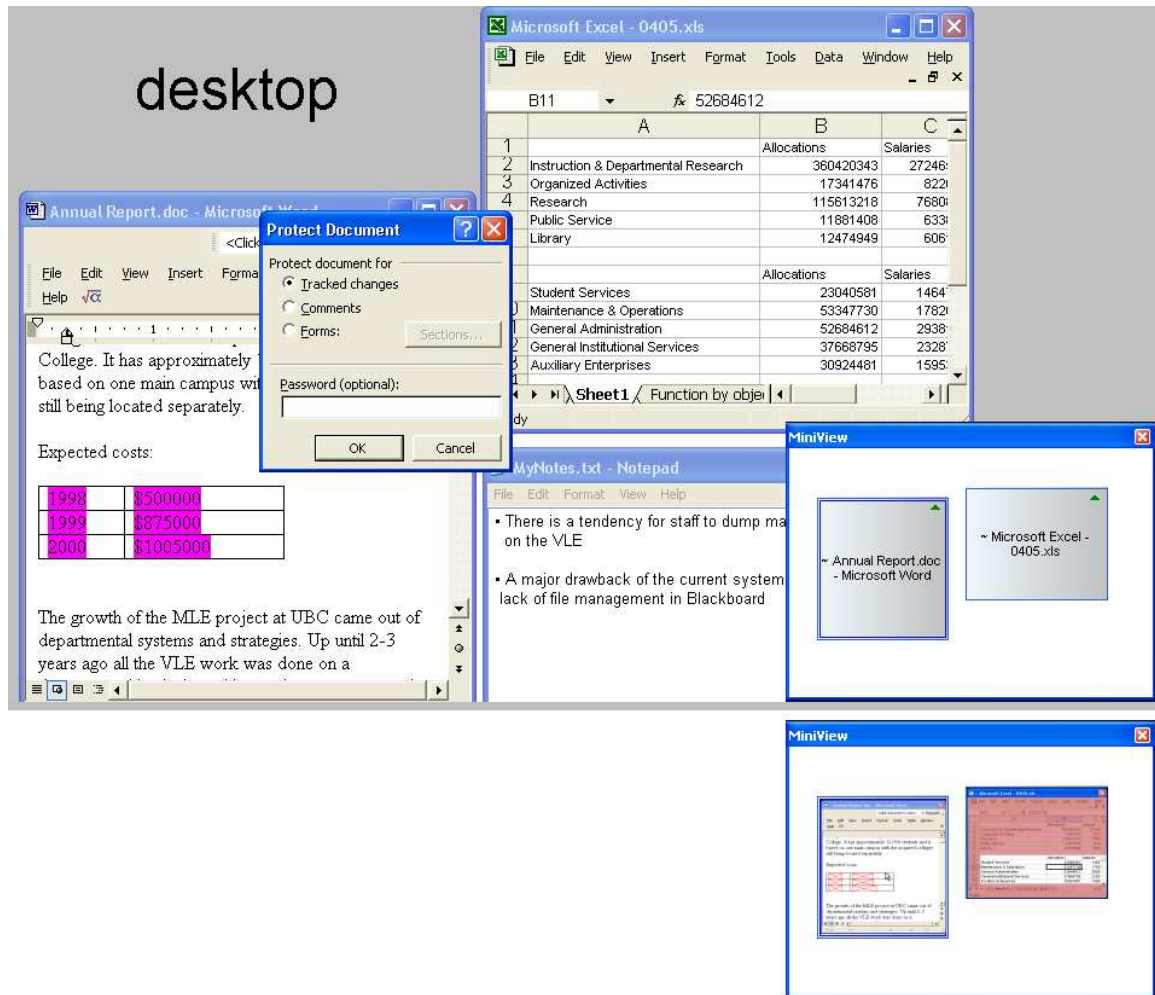


Figure 4.11: Radar View: In this case a Word document and an Excel spreadsheet are shared on the public view. Other controls and documents on presenter's desktop (such as personal notes or a document protection dialog) are private. The radar view provides awareness for the presenter on which windows are shared. Alternatively, it can also show miniatures of the public view with the blurred or highlighted regions.

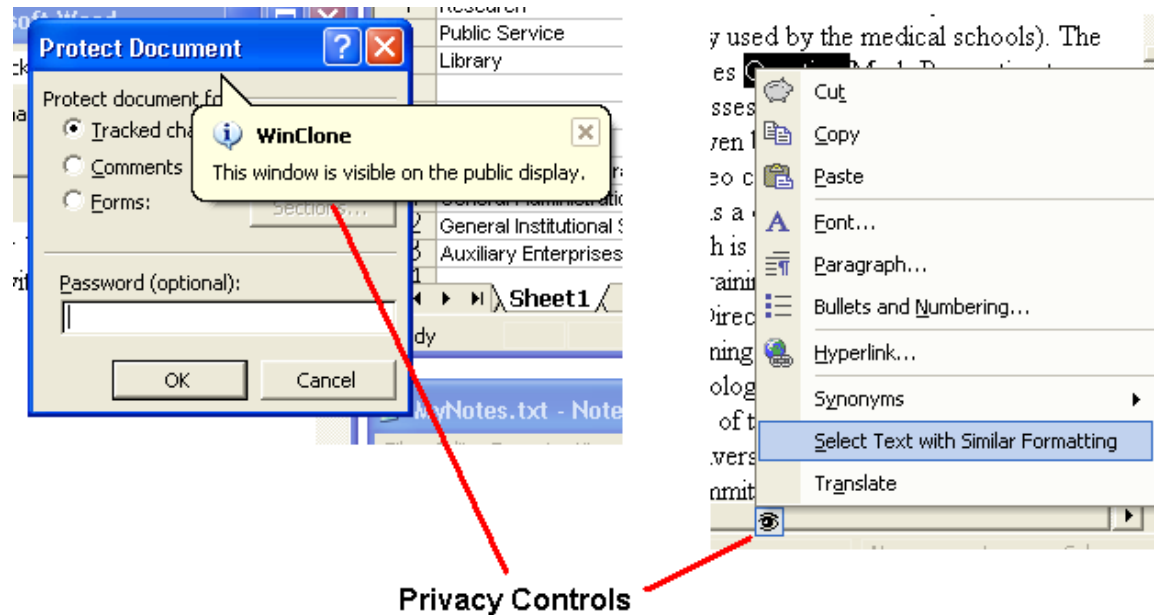


Figure 4.12: A popup notifier that appears when an application dialog is opened (left) can remind presenter whether the audience can see this dialog or not according to the active policy. It can also provide a control for bypassing the policy (not implemented in the prototype). A less obtrusive privacy control relies on adding a mini icon window that “floats” next to private UI components (e.g. attached to a popup menu as seen on the right). It provides a privacy indication for the presenter and if the mouse hovers on it long enough will bypass the privacy policy and expose the menu on the public screen.

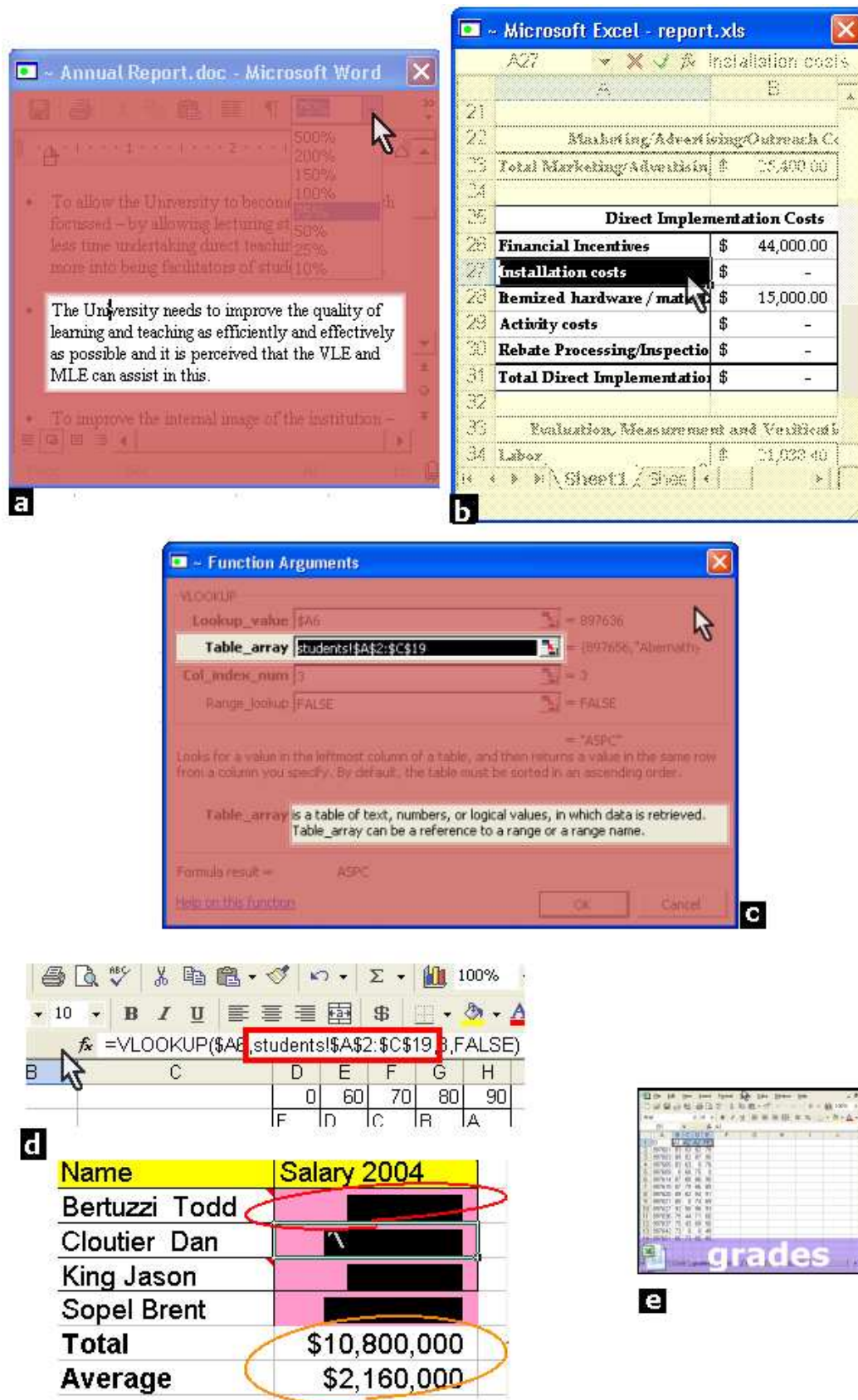


Figure 4.13: Auto-highlighting of active context: (a) active paragraph is highlighted, (b) outlining the table surrounding selection, (c) highlighting active dialog field; Auto-highlighting changes: (d) “by-hand” style circling of changed formula parameters, changed cells and dependent cells, and (e) highlighting sheet switching in a downsized view (compare with Figure 3.4a).

Chapter 5

User Study

The system that has been developed is a working prototype for the architecture and design presented in this thesis. This chapter describes an experimental user study whose purpose was to assess the degree to which the system meets its design goal of supporting viewers in a generalized presentation.

When considering an evaluation of our system we should note that it is required to measure the system's effect on both the presenter and the viewers to gain a full understanding of the effectiveness of the system. For a presenter the question is whether the system improves her ability to maintain her privacy and be more comfortable in making the presentation. One aspect of this is which of the privacy filters are better. For viewers, a primary question is whether the privacy-related manipulations diminish the viewers' ability to understand the presenter's actions. A related question is whether the augmentations targeted to passive viewers assist an audience in following the presenter.

We decided to focus on the second set of questions (utility for viewers). The presenters' perceived privacy is a very subjective measure. One of the problems in a controlled experiment is that a presenter may not have the same privacy sensitivities for mocked-up data that would exist for her own data. Therefore, a true evaluation of this part of the system may be best achieved in a field study, where presenters will be using it for their real-world presentations. This would allow us to assess the subjective effectiveness of the system as it is experienced by presenters. A secondary reason is that we already know that a person engaged in a presentation is too busy to notice and prevent privacy leaks. Therefore, we can be somewhat confident in assuming that the ability to define privacy concerns beforehand and have them automatically handled is at least partially beneficial for presenters. Our own experience bears this out, although it would of course be important to confirm this with a formal study.

The initial scope of the experiment focused on the effects on viewers of the privacy

filters. However, it also examined the effects of visual augmentations because this could be measured at the same time and interactions between the two were possible. As we will show later in this chapter, the interaction between the two manipulation types did turn out to be of particular interest.

5.1 Methodology

To assess the system's effect on passive viewers, we chose to use a training scenario as an example of a generalized presentation. A passive viewer (subject) is trained on a set of tasks, watching presenter's interactions with an application on a public display. This allows the shared view to be manipulated and altered using one or more of the techniques introduced in Section 4.2 throughout the training. After being trained a viewer is asked to complete a similar task on his own. It is then possible to assess the subject's **performance** from different aspects (speed, accuracy, efficiency). The monitored parameters and the tools to collect them will be described in detail in Section 5.2.

We chose an Excel spreadsheet as the shared application and a set of tasks common to a real-world grade report preparation for use in the study. This choice allowed the experiment to have a reasonable level of external validity. In addition Excel has a vast collection of features and functionalities to which we can apply our techniques. Some features are familiar to most users and some are not, so a combination of these can be used for testing. In addition most Excel features we included in the tasks have equivalent counterparts in other applications.

We describe a wide range of manipulations to the shared application view. Some are privacy enhancements aimed at maintaining the presenter's privacy and others are visual augmentations for passive viewers. We will refer to the former as *privacy filters* and the latter as *augmentation effects* or *filters*.

Therefore, we have two factors we can control: Privacy and Augmentation. In the experiment a coarse split to two levels, "on and off" was used. The "on" level was mapped to a subset of manipulations. The following is a list of the possible four conditions and the manipulations they entail:

N - Normal view – this is the control condition, showing regular screen recordings without any manipulations (as one might get from using VNC). We will refer to this con-

dition as **R**(regular) or **N**(normal) in the analysis

P - (Privacy) – blurring or excluding private information on the public view.

- Blurring specific ranges of cells
- Replacing file selection dialogs with animated icons
- Blurring specific menu items (recent files and others)
- Auto-Hiding error dialogs

A - (Augmentation)

- Highlighting of active context
- Highlighting active dialog field
- Highlighting of changes
- Visual indication on mouse clicks
- Report keyboard shortcuts as subtitles
- Extended “decay” period for menus and dialogs (+ highlighting selected menu item)
- Replacing frequent context menus with subtitles
- Replacing file selection dialogs with animated icons
- Replacing a wizard with an iconic animation
- Auto-Hiding error dialogs

PA - (Privacy + Augmentation) The union of the features of the A and P conditions.

Some of the features can be considered both as privacy preservers and as augmentation for passive viewers. For example replacing file selection dialogs with iconic indicators does not expose sensitive file system views on the one hand and mitigates visual clutter on the other hand.

We were particularly interested in the interaction between the privacy and augmentation factors. Does privacy have different effects when used with or without the combination of

visual augmentations?

The hypotheses we were trying to test in the study were:

Hypothesis #1 – Adding visual augmentations in the training will affect subjects' task performance (conjectured improvement)

Hypothesis #2 – Adding privacy filters in the training will affect subjects' task performance (conjectured degradation)

Hypothesis #3 – The effect of privacy depends on the the additional use of visual augmentations (P X A - Interaction Effect)

Hypothesis #4 – The overall ordering of task performance for the training conditions will be: $f_{pm_i}(P) < f_{pm_i}(N) < f_{pm_i}(PA) < f_{pm_i}(A)$ ¹

These hypotheses led to the following testable null hypotheses:

H_01 : Performance is not affected by adding visual augmentations in the training

H_02 : Performance is not affected by adding privacy filters in the training

H_03 : The effect of privacy filters does not depend on the presence of visual augmentations

H_04 : All four different training types will lead to the same performance levels

5.1.1 Experimental Design

We used a $2 \times 2 \times 4$ mixed model design with privacy and augmentation being between subject factors (with two levels each - on and off) and task being a within subject factor with four levels as will be described later. We used a set of dependent variables to measure performance (speed, accuracy and efficiency) that will be described in Section 5.3.1.

The main statistical test used for this model was a mixed model factorial ANOVA (Analysis of Variance) on the entire design. The significance level for all tests was chosen to be 0.05.

¹ $f_{pm_i}(\cdot)$ denotes a primary performance measure from the set: speed, efficiency, accuracy as described in Section 5.3.1.

5.2 Method

In the experiment participants were asked to complete a grade report template in Excel (see Appendix B) after watching recorded training movies with different visual augmentations.

5.2.1 Participants

Twenty-eight subjects participated in the experiment. Six were female and twenty-two were male. Seven participants were randomly assigned to each one of the four between subject conditions (R,P,A,AP). Subjects were paid \$10 for their participation.

Twenty six participants were undergraduate or graduate students in the Department of Computer Science or the Department of Electrical and Computer Engineering at the University of British Columbia. Two participants were graduate students in a different department at the same university.

Previous Excel experience

All participants had basic familiarity with Excel or an equivalent spreadsheet application. Four subjects (14.3%) use it once a week, eighteen subjects (64.3%) use Excel on a monthly basis, and four subjects (14.3%) use it less frequently.

All subjects ranked themselves as having basic or intermediate overall expertise with Excel or a different spreadsheet (20 subjects ranked themselves as having minimal or basic expertise and 8 subjects ranked themselves as intermediate, see Figure 5.1). Subjects were specifically screened for not being Excel experts (i.e. using Excel on a daily basis and familiar with most functions).

Subjects were also asked to rate their expertise with the specific Excel features used in the tasks. These reports were collected after the experiment to support the analysis and are summarized in Table 5.1.

5.2.2 Instruments and Data Collection

The following is a description of the materials and monitoring tools used in the study.

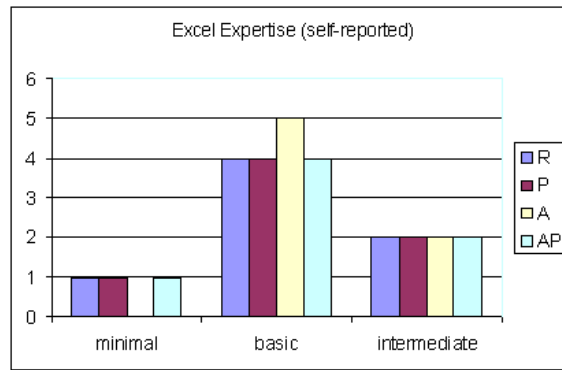


Figure 5.1: Subject Excel Expertise: The four conditions were on par (even though subjects were randomly assigned to conditions).

Functionality	R			P			A			AP		
	U	B	E	U	B	E	U	B	E	U	B	E
Multiple sheets	0	7	0	0	4	3	0	6	1	2	3	2
Import data	4	2	1	3	4	0	5	2	0	2	4	1
lookup	7	0	0	6	0	1	7	0	0	5	2	0
conditional format	5	2	0	6	1	0	6	1	0	6	1	0
SUMPRODUCT	6	1	0	4	3	0	7	0	0	5	2	0
ROUND	3	4	0	2	3	2	2	5	0	4	3	0
Absolute refs	4	3	0	3	2	2	4	3	0	6	0	1
Table sorting	2	4	1	2	3	2	0	7	0	0	6	1

Table 5.1: Subject familiarity with Excel functionality – U=unfamiliar, B=basic and E=expert. There is one notable difference between conditions with respect to the use of absolute references. This functionality was the core of Task 2 and some of Task 3 and subjects in condition AP had less previous experience with it compared to the other conditions.

Training movies

To make all training sessions as comparable as possible, apart from the controlled factors, we chose to pre-record these as screen capture videos rather than conducting live training sessions (which may differ from subject to subject).

We used our system to create a manipulated view of interactions with Excel (with one of the N,A,P or AP feature sets) and recorded this view using an off-the-shelf screen capturing tool. The same soundtrack was used for all training movies. Altogether twelve movies were created: 4(conditions)×3(subtasks).

The movies were played back for subjects using a movie player that supports a true full screen mode (no UI controls or other artifacts were present on the screen while the movies were being played).

Questionnaire

At the end of the experiment session participants were asked to complete a short questionnaire (Appendix ??). The questionnaire comprised four parts:

- Past Excel experience - Participants were asked to rank their frequency of using Excel and their expertise level.
- Evaluation of the specific Excel functionalities used in the task: (i) Prior familiarity (not familiar, basic and expert) and (ii) Training session effectiveness for the specific functionality on a 5-point scale (1=Highly ineffective, 5=Highly effective)
- Training session and task experience – eleven statements about the quality of the training and the ease of performing the tasks. Participants were asked to rank each statement on a 7-point Likert scale (1=Strongly Disagree, 7=Strongly Agree)
- Effectiveness / Disturbance of the specific visual enhancements and privacy filters – participants ranked eleven augmentation features on a 5-point scale (1=Disturbing, 5=Effective) and a N/A option if the specific feature did not appear in the training.
- Free form comments and suggestions

Excel-Logger

To monitor subject performance a special logging tool, ExcelLogger, was created (written in C#). We utilized the monitoring techniques from our system (Excel API hooks, Accessibility and OS hooks) to provide us with the following logging information:

- Excel user interactions (cell changes, selection and worksheet changes).
- Menu selections (time to selection, aborted menus) and dialog interactions.
- Use of Excel's help wizard.
- Mouse clicks and keyboard presses (including shortcut keys)

We automated a screen recording component² to work in concert with the logger, so video recording of the subject interactions were also created (and the logged events can be used as an index)

The ExcelLogger tool also functioned as a wizard that led each subject through the tasks and training components (see procedure section)

A log analyzing component was programmed to parse the interaction logs and produce different aggregated measurements that served as the dependent variables as will be described in Section. 5.3.1.

5.2.3 Procedure

Each session lasted an hour to an hour and a half. Participants first watched a short movie explaining about the different augmentation effects and privacy filters (about two minutes long). They then watched a series of three short subtask training movies (four to six minutes long each). After each subtask training movie subjects were asked to complete the subtask on their own in Excel. They were provided with written notes summarizing the training and the steps they were required to reproduce.

The first subtask focused on setup and importing data from external data files into designated spots in the report, the second subtask focused on the use of lookup formulas and absolute references to connect data from different sheets, and the third subtask combined various functionalities such as grade computation formulas, conditional cell formatting and data sorting options.³

After completing the three subtasks, subjects were asked to complete the entire grade report filling task on a new Excel worksheet with different course data. No training movie was provided for this task, so subjects were forced to use the knowledge they retained from the previous training. For all tasks subjects were allowed to use Excel's help wizard if they needed to. If they failed to resolve a problem using Help, they could ask for the experimenter's intervention (these interventions were logged as well).

All subjects had the same set of tasks. All three training movies were similar (apart from the set of augmentation and privacy filters used in the movie).

²CamRecorder.exe - see Camtasia Studio Online Help.
http://download.techsmith.com/camtasiastudio/docs/onlinehelp/studio_help.pdf (page 204) (last checked August 18th, 2005)

³A full description of the steps required to complete each task can be found in Appendix B.

We ran a series of pilot studies to tune the tasks. An early pilot showed us that providing a single training session on the entire task did not work. Subjects did not remember what they saw in the training and relied instead on Excel's help. On the other hand, we felt that a single shorter task was too susceptible to individual differences. We decided that breaking the training and tasks into smaller units fit with pedagogical guidelines and allowed better testing of the training effect on performance.

At the end of each session the participant was asked to complete a short questionnaire and was then given a debriefing.

5.3 Results

5.3.1 Measuring Performance

We used measures of three different dependent variables as primary indicators of performance for each one of the tasks:

- **Speed** - Overall time to completion (TIME)
- **Accuracy** - The resulting report from each task was assessed using a marking scheme and was assigned a floating point grade in the scale 0-1. (GRADE)
- **Efficiency** - Overall number of actions to complete the report (#ACTIONS) as recorded by the ExcelLogger (Section. 5.2.2).

We also carried out an analysis of secondary dependent variables that can assist in establishing an understanding of the different factor effects.

- Excel operations – breaking down the total number of actions based on Excel operations: number of times cell content was changed (#CC), number of times cell selection was changed (#SC), number of times sheets were switched (#SW) and number of times “undo” was used (#UND). It is reasonable to assume that a subject who commits more errors on the path to the solution or is not sure how to complete the task will require more of these Excel operations. Therefore, these counts better reflect efficiency and to some extent provide an indication of the the number of “on-the-fly” errors (as opposed to the final number of errors reflected in the grade or the overall number of actions that includes many other event types).

- Menu operations - Two count measures were taken: the number of menu selection (#MS) and (#MA), the number of aborted menus (a menu that was opened and closed without committing a selection).

An additional time measure was also computed: average first menu selection time (FMST). For each menu item selected in all of the tasks, only its first selection time was taken into account, which reflects the effort of finding the menu item.

To capture the overall search time, which may include opening and closing of other menus, the selection time was defined as: the time from the first detection of an opened menu window until selection was made. Aborted menus followed by the opening of a new menu within a certain time gap (10 second) were considered to be the same operation.

- Help Requests - counting the number of help uses or interventions may be misleading, since separate requests can still refer to the same problem. Therefore two separate measures were used: a boolean value indicating whether help or interventions were required (HELP and INTERVENTION) and the total time the help window was used (HELP-T).
- Shortcut key usage – the absolute number of shortcut keys that were used is misleading (for instance, a subject who committed a lot of errors may use these more). Therefore, a boolean value was used, one for the cell filling operations (CTRL: Ctrl+D, Ctrl+R) and one for the absolute references toggling using the function key four shortcut (F4) that were demonstrated in the training.

5.3.2 Quantitative Analysis

We performed a statistical analysis of subject performance, based on the data collected by ExcelLogger.

Primary performance measures

Table 5.2 summarizes the results of the global repeated measures ANOVA.

Dependent Measure	Factor	Significance
Speed (Time)		
	Task	$F_{3,72}=60.669$, p=0.001 (partial $\eta^2 = .717$)
	P	$F_{1,24}=1.059$, p=.314
	A	$F_{1,24}=0.196$ p =.662
	A*P	$F_{1,24}=1.106$, p=0.303
Efficiency (# Actions)		
	Task	$F_{3,72}=37.177$ p=0.001 (partial $\eta^2 = .608$)
	P	$F_{1,24}=0.99$, p=.756
	A	$F_{1,24}=0.186$ p =.670
	A*P	$F_{1,24}=1.475$, p=0.236
Accuracy (Grade)		
	Task	$F_{3,72}=10.809$ p=0.001† (partial $\eta^2 = .311$)
	Task*A*P	$F_{3,72}=3.459$, p=0.05† (partial $\eta^2 = .126$)
	P	$F_{1,24}=0.596$, p=.448
	A	$F_{1,24}=0.437$ p =.515
	A*P	$F_{1,24}=5.071$, p=0.034 (partial $\eta^2 = .174$)

Table 5.2: Testing for performance differences among the four conditions. Mauchly’s test of sphericity was significant for the entries indicated by a †. Therefore, the Greenhouse-Geisser correction was taken into account in these entries, so the F statistic was computed with adjusted degrees of freedom $F_{1.649,39.579}$.

Task effect

The repeated measures $2 \times 2 \times 4$ ANOVA showed a statistically significant task effect for all primary measures (TIME, #ACTIONS, GRADE).

However, the differences in time and number of actions due to the task are not interesting. The tasks were not designed to be equal (especially the fourth one, which is a composition of the other three).

The statistically significant differences in grades between tasks (Figure 5.2) are again not surprising because the tasks were different. It does indicate that the tasks were not equal in difficulty.

Running post hoc paired samples t-tests on the overall grades we can verify that Task 2 had significantly lower grades than the other tasks ($p \leq 0.02$). The other tasks were not significantly different in grades apart from Task 3 and Task 4 ($p \leq 0.024$).

The common errors in the tasks were: forgetting to fill in parts of the cover page or copying wrong parameters (Task 1), wrong use of absolute referencing, inaccurate parameters for the lookup functions or simply copying values instead of using the shown formulas (Task 2 and Task 3). In Task 4 similar errors were observed.

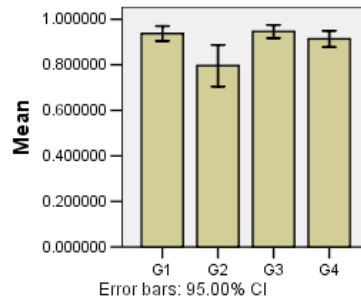


Figure 5.2: Average grade per task

Effects on speed and efficiency

No significant effects for Augmentation, Privacy or interactions were found with respect to TIME and #ACTIONS in the global ANOVA or in two-way factorial ANOVAs (PXA) conducted for each task separately.

Figures 5.3 and 5.4, summarize the results for these two measures. While no effect was statistically significant, it seems that adding augmentation alone or privacy alone (A or P) had some consistent improvement on time and efficiency. Adding augmentation and privacy together (AP) varied between tasks and measures and in some cases degraded performance.

It is also evident that there was great variation between subjects (both globally and in each condition) with respect to speed and efficiency. These variables were probably more related to the overall expertise a subject had with Excel and his interaction style, neither of which are likely to change much after only a short training session.

Understanding interactions for grades

There was a statistically significant $P \times A$ interaction and only borderline significance for a $TASK \times P \times A$ interaction (Table 5.2). These were further analyzed.

$P \times A$ is an interesting interaction as can be seen in Figure 5.5. The analysis was followed by Post-Hoc LSD tests.

Condition P had significantly greater accuracy ($p < 0.043$) than the control condition, as did Condition A, though with only borderline statistical significance ($p < 0.05$). No other differences were significant (it should be noted that these differences were not significant with more conservative tests such as Bonferroni or Tukey's HSD).

In other words, adding privacy filters without augmentation or augmentation without privacy increases overall accuracy, whereas adding privacy together with augmentation does

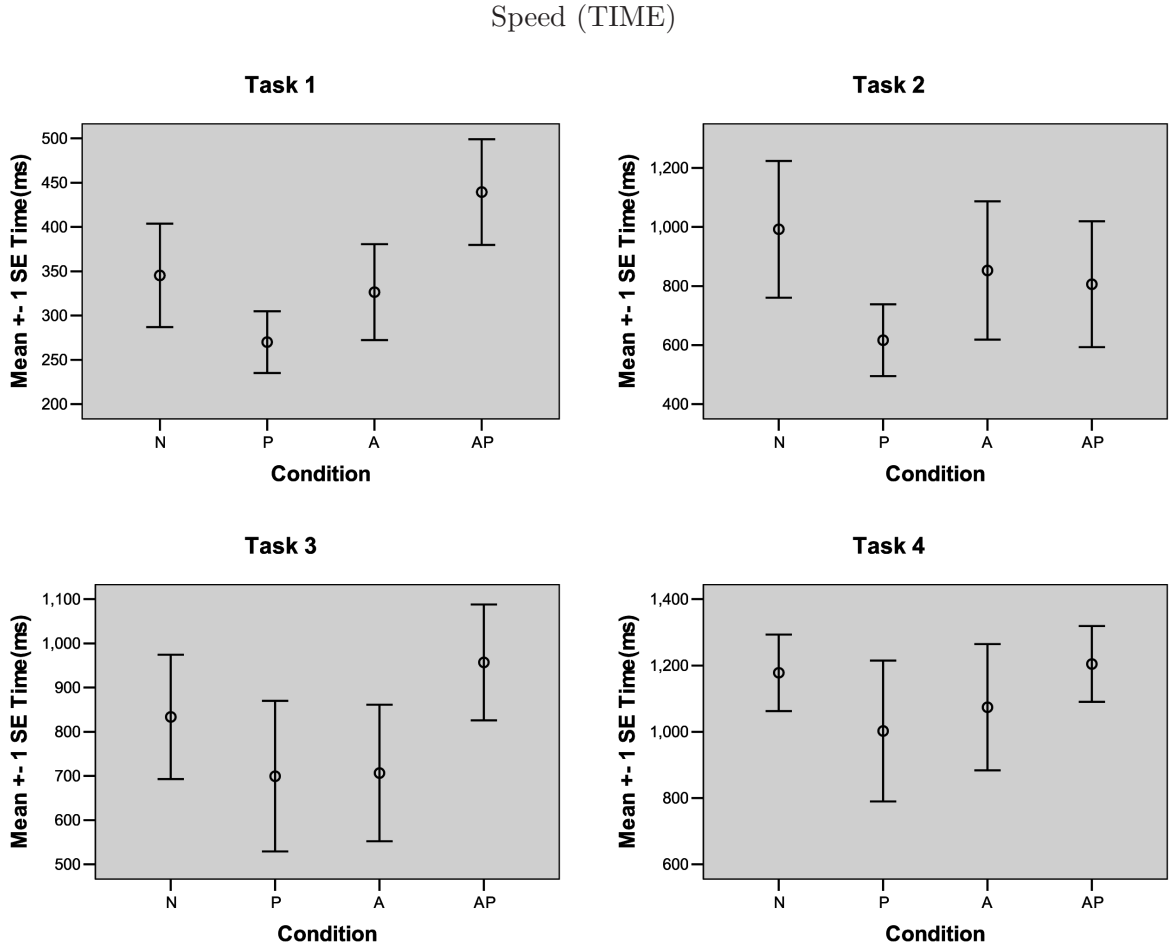


Figure 5.3: Effects of condition on speed: None of the effects was statistically significant at the 0.05 level.

not increase accuracy (in fact it degrades accuracy as can be seen from the graph, but the difference is not statistically significant).

To understand the $\text{TASK} \times \text{P} \times \text{A}$ interaction, separate two-way ANOVAs (PXA) were conducted for each task. Figures 5.6 and 5.7 show the per task effects. Although Tasks 1,3 and 4 seem to have the same behavior, there are no significant main effects nor $\text{P} \times \text{A}$ interaction effects for Tasks 1 and 4.

Task 3 had a statistically significant main effect of A ($F_{1,24}=4.289$, $p \leq 0.049$, partial $\eta^2 = .152$). Subjects who had visual augmentations for Task 3 had greater accuracy than subjects who did not (Mean(A=1)=0.973, SD(A=1)=0.045, Mean(A=0)=0.919, SD(A=0)=0.089). No significant P effect or interaction was detected.

Task 2 had a statistically significant $\text{P} \times \text{A}$ interaction effect ($F_{1,24}=5.207$, $p \leq 0.032$, partial $\eta^2 = .178$), but no significant main effects. Post-Hoc LSD tests show that there

Efficiency (# Actions)

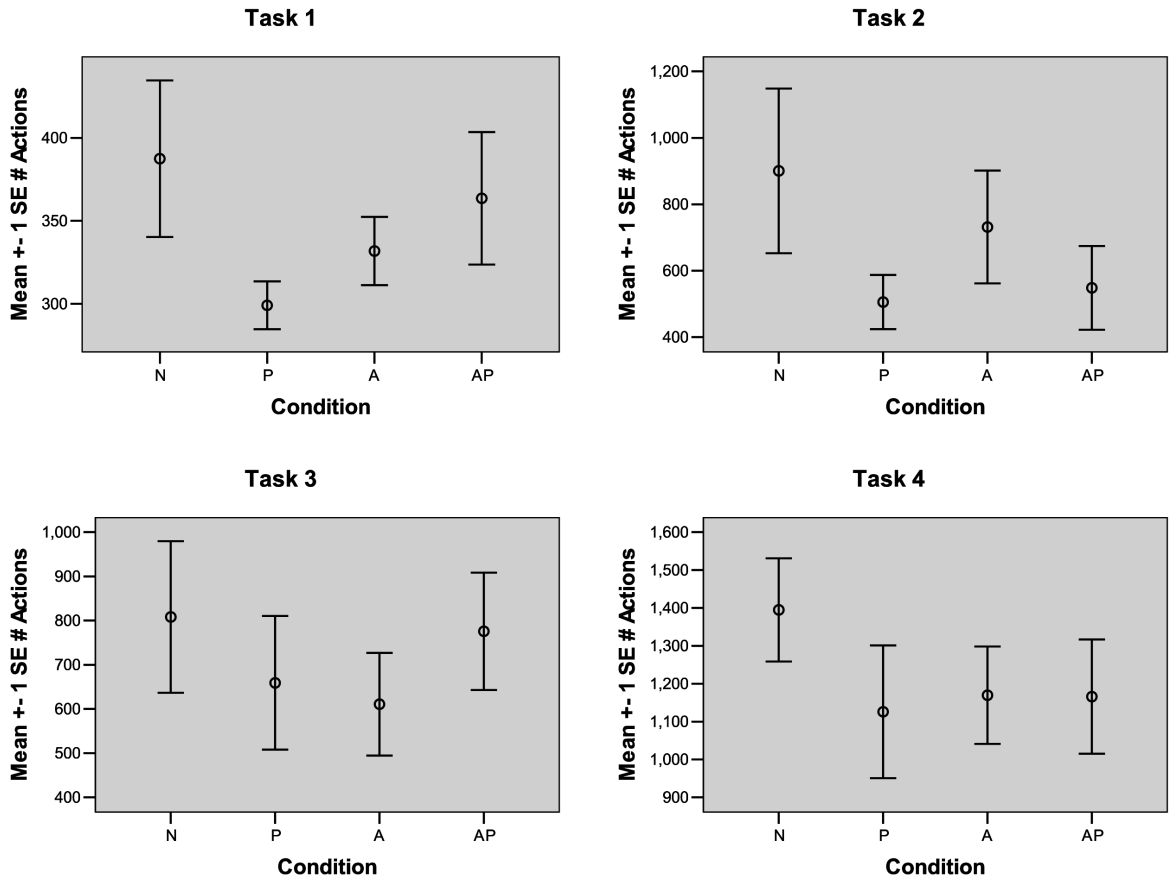


Figure 5.4: Effects of condition on efficiency: None of the effects was statistically significant at the 0.05 level.

was a statistically significant difference between the control condition (R) and P for this task ($p \leq 0.042$). Subjects in condition P were more accurate than subjects in the control condition (Mean(P)=0.928, SD(P)=.095, Mean(R)=0.671, SD(R)=.275). No other simple main effects for this task were detected.

The lack of significant effects for Task 4 indicates that in terms of learned functionality retention, all conditions behave more or less the same. This was also verified by running $2 \times 2 \times 3$ ($P \times A \times \text{SUBTASK}$) ANOVA on the accuracy differences between each original subtask and its equivalent part in Task 4 (δ_{grade}). This ANOVA showed only a significant interaction $\text{TASK} \times A$ ($F_{1,1379,33.092}^4=5.207$, $p \leq 0.033$, partial $\eta^2 = .154$). Additional one-way ANOVAs for each subtask on A showed that for Tasks 1 and 3 there was no significant A effect. For Task 2 there was a significant effect for A ($F_{1,26}=4.342$, $p \leq 0.047$, partial $\eta^2 =$

⁴Greenhouse-Geisser correction

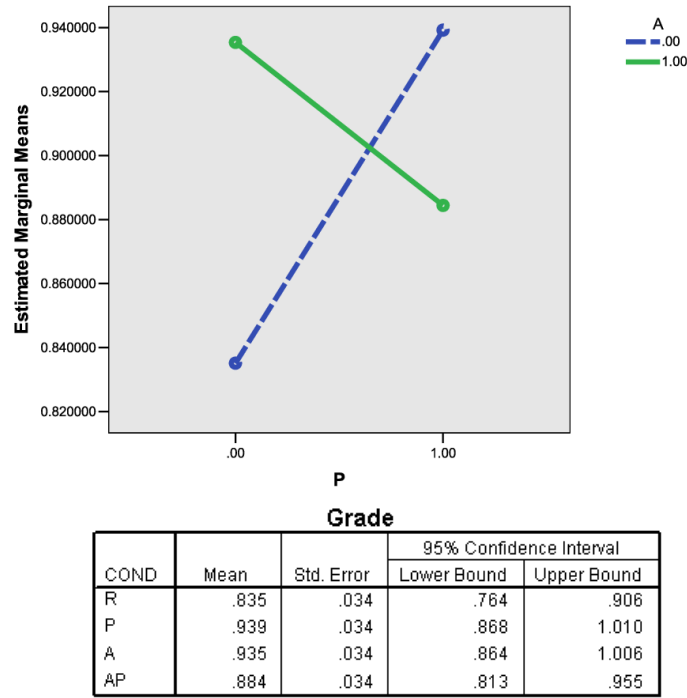


Figure 5.5: $P \times A$ interaction (across all tasks). when $A=0$ adding privacy improves accuracy (statistically significant), whereas when $A=1$ adding privacy reduces accuracy (not significant).

.143), participants with augmentations improved a little bit in Task 4 ($Mean_{A=0}(\delta_{grade}) = -0.0054$, $SD_{A=0} = .115$, $Mean_{A=1}(\delta_{grade}) = 0.091$, $SD_{A=1} = .130$). For Task 3 the overall accuracy dropped a bit (not significant), perhaps due to fatigue effects.

Secondary performance measures

We also performed a number of statistical tests on the secondary measures:

Excel operations - The same $2 \times 2 \times 4$ repeated measures ANOVA was used, with the separate dependent variables being: #CC, #SC, #SW and #UND.

Apart from a non-interesting significant task effect for all variables, the only statistically significant difference found was a main effect of P on #SC, the number of cell selection changes, ($F_{1,24}=5.001$, $p \leq 0.035$, partial $\eta^2 = .172$). Subjects who had privacy filters committed fewer cell changes ($Mean(P=0)=113.446$, $Mean(P=1)=77.589$, $SE=11.338$).

Menu operations - The same ANOVA was conducted on #MS and #MA, but no

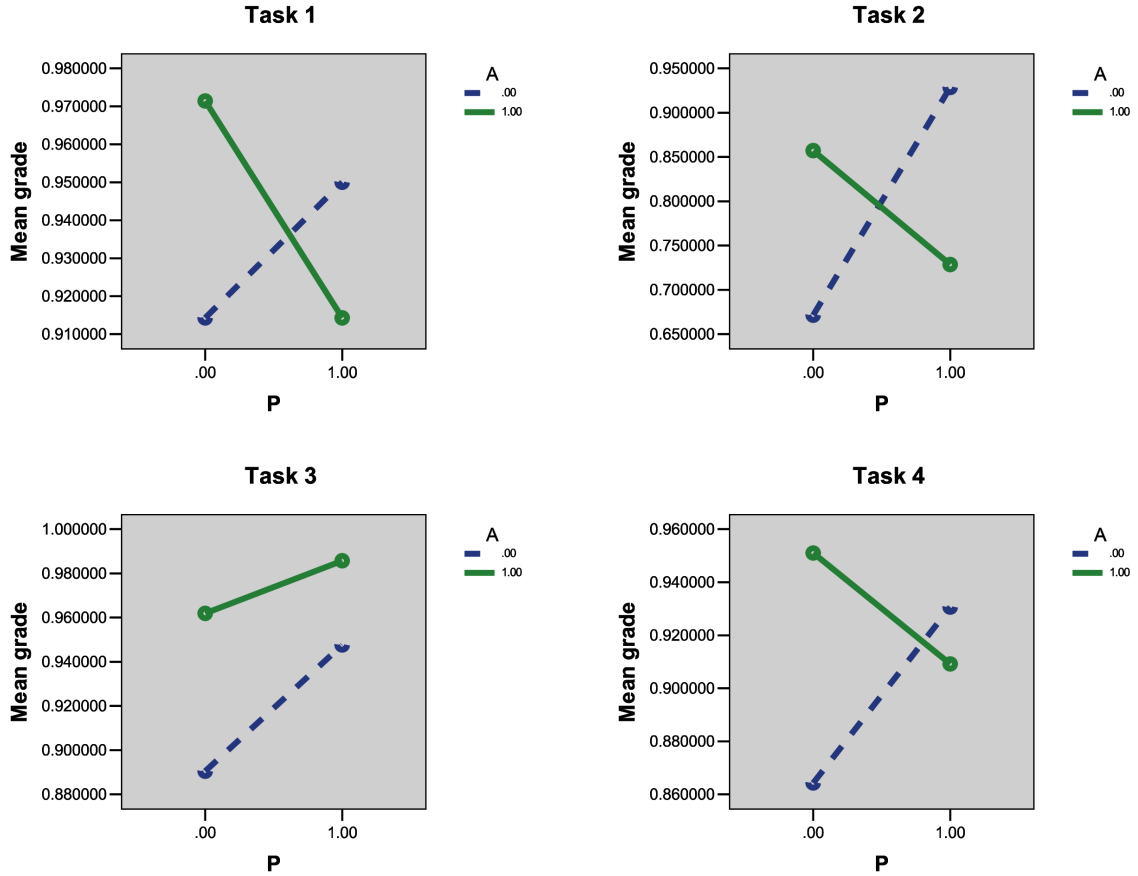


Figure 5.6: $P \times A$ interaction per task. Task 3 has a significant main A effect and no interactions. Tasks 1 & 4 have no main effects and no interactions and Task 2 has a significant $P \times A$ interaction.

statistically significant effects or interactions were found, apart from a task effect (not interesting).

To test FMST, which was a cross-task measure, a factorial 2×2 ($P \times A$) ANOVA was used. The test detected only a main effect of A ($F_{1,24}=4.815$, $p \leq 0.038$, partial $\eta^2 = .167$). Participants who had the visual augmentations spent about one second less on average when looking for menu items for the first time (Mean(A=0)=3862.571 ms, Mean(A=1)=2709.214 ms, SE=371.661).

Keyboard shortcut use - CTRL and F4 were two boolean dependent variables aggregated over all the tasks. A non parametric Kruskal-Wallis on independent samples test was used with three different grouping possibilities (A,P, $P \times A$). Statistically significant differences were detected with respect to A for both variables.

CTRL: $\chi^2(df = 1) = 3.947$, $p \leq 0.047$, only 7 out of 14 subject who did not have visual

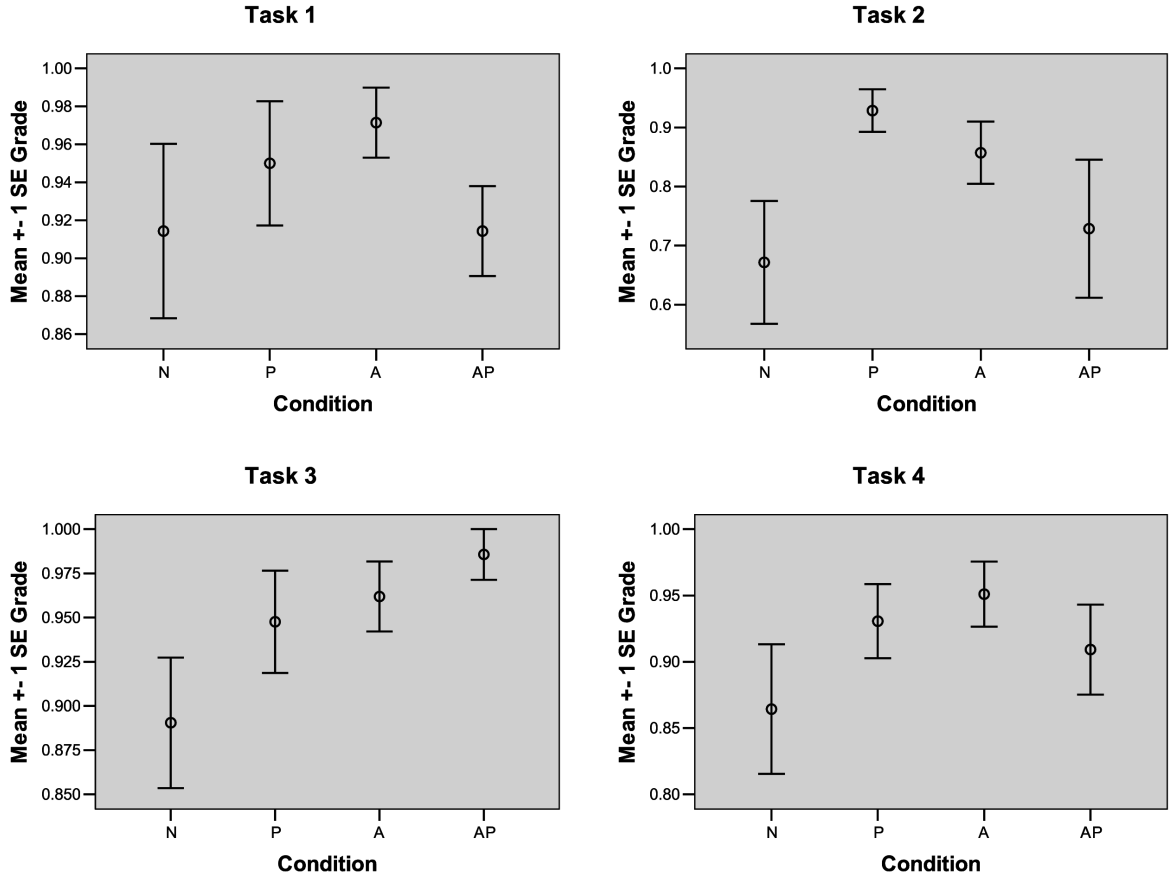


Figure 5.7: Condition effects on accuracy per task: The grades for each condition per task (alternative representation to the one presented in Figure 5.6 that displays standard deviations as well).

augmentations used the Ctrl-R and Ctrl-D shortcuts, as opposed to 12 out of 14 subjects who used them when augmentations were employed in the training.

F4: $\chi^2(df = 1) = 5.400, p \leq 0.020$, only 6 out of 14 subject who did not have visual augmentations used the F4 shortcuts, as opposed to 12 out of 14 subjects who used it when augmentations were used in the training.

Testing for subjects who used both types of shortcuts also showed a strong effect for augmentation ($\chi^2(df = 1) = 8.816, p \leq 0.003$) with only 3 out of 14 subject in A=0 using both shortcut types, as opposed to 11 out of 14 subjects in A=1.

Help Requests - A $2 \times 2 \times 4$ repeated measures ANOVA was conducted on HELP-T (total help window use time), but no statistically significant effects were detected (apart from task).

A non parametric Kruskal-Wallis on independent samples test was used on HELP (boolean indicating if subject used help overall) and INTERVENTION (boolean indicating if subject asked for experimenters help overall) and the combined “HELP or INTERVENTION” variable. The latter may be more reflective of subject problems, since some subjects solved the problems using HELP and did not need intervention (subjects were instructed to try to solve their problem using help first).

The test was run using the three different grouping possibilities (A,P,P \times A). No statistically significant differences were detected on HELP or INTERVENTION. The “HELP or INTERVENTION” variable did have a statistically significant A effect, ($\chi^2(df = 1) = 3.947, p \leq 0.047$), with 12 out of 14 subjects in A=0 requiring help as opposed to 7 out of 14 in A=1. It should be noted, however, that in Tasks 2 and 3 more subjects in A=1 asked for intervention than A=0 but without statistical significance (5 vs. 4 in Task 2 and 3 vs. 1 in Task 3).

Most help usage and some interventions revolved around the use of absolute references in the lookup formulas and the structure of the lookup formula.

Some of the requested interventions were around an unexpected technical problem (Excel “saved” the import data as queries and was auto-completing the lookup formula parameters with these). And a few others requested interventions were regarding a confusion between the two data sets used for the tasks.

5.3.3 Questionnaire Analysis

We performed a qualitative analysis of the data collected via the questionnaire (Appendix ??).

Training session and task experience

We used the non-parametric Kruskal-Wallis test to detect differences on the 7-point Likert scale rating medians between subject groups (it was separately used on the A,P and A*P factors). The only significant differences detected were with respect to factor A on two questions (no significant effect for P or P \times A interactions):

- Q7: “Overall it was easy to complete the task”, $\chi^2(1) = 4.268, p < 0.05(0.038)$, (Med(A=0)=5, Avg(A=0)=4.857, SD(A=0)=1.231 ; Med(A=1)=6, Avg(A=1)=5.785, SD(A=1)=1.477)

- Q10: “It was easy to replicate the presenter’s actions”, $\chi^2(1) = 4.572, p < 0.05(0.033)$,
(Med(A=0)=5 , Avg(A=0)=4.571, SD(A=0)=1.158 ; Med(A=1)=6 , Avg(A=1)=5.5,
SD(A=1)=1.557)

We found some notable differences (though not-significant) with respect to the training and task completion experience (graphs in Figure 5.8).

“Negative statements” ratings: - Subjects who had visual augmentations (A=1) expressed more concern about the amount of detail in the training (especially A=1,P=0), but at the same time found it less difficult to follow the training. Subjects who had augmentations were much more divergent in their opinions than subjects who did not have such augmentations (Q3: SD(A=0)=0.938, SD(A=1)=2.336 and Q4: SD(A=0)=1.447, SD(A=1)=2.277).

Participants in condition A were much more certain (almost unanimously (Q6: Med(A)=1, SD(A)=0.787) that there was no missing information in the training, as opposed to conditions R and P that were more diverse and a bit less certain (Q6: Med(R)=2, SD(1)=1.864, Med(P)=2, SD(P)=2.478). Especially interesting are subjects in condition AP who had a notable bi-polarity (four subjects gave ratings of 5 or 6 and three subjects rating of 1).

“Positive” statements ratings: -

Subjects from conditions P, A and AP expressed a quite uniform rating for training pace (Q5: Med(A,AP)= 5, Med(P)=6). Subjects who were in the control group (R) were polarized (4/7 gave 5 or 6 ratings and 3/7 gave 2 or 3 ratings). A similar pattern can also be seen with respect to Q8. Subjects in conditions P,A and AP feel quite comfortable using the Excel techniques they were trained on (Q8: Med(P,A,AP)= 6), while subjects from the control condition have more diversity (3/7 rated 4, 2/7 rated 6 and 2/7 rated 7).

A different pattern can be seen for the overall “pleasant to watch” rating (Q11). The median ratings for all conditions are equal (apart from AP which is a bit lower). However, subjects who had privacy (P,AP) were more uniform with their rating as opposed to subjects without privacy (R,A) some of whom gave lower ratings.

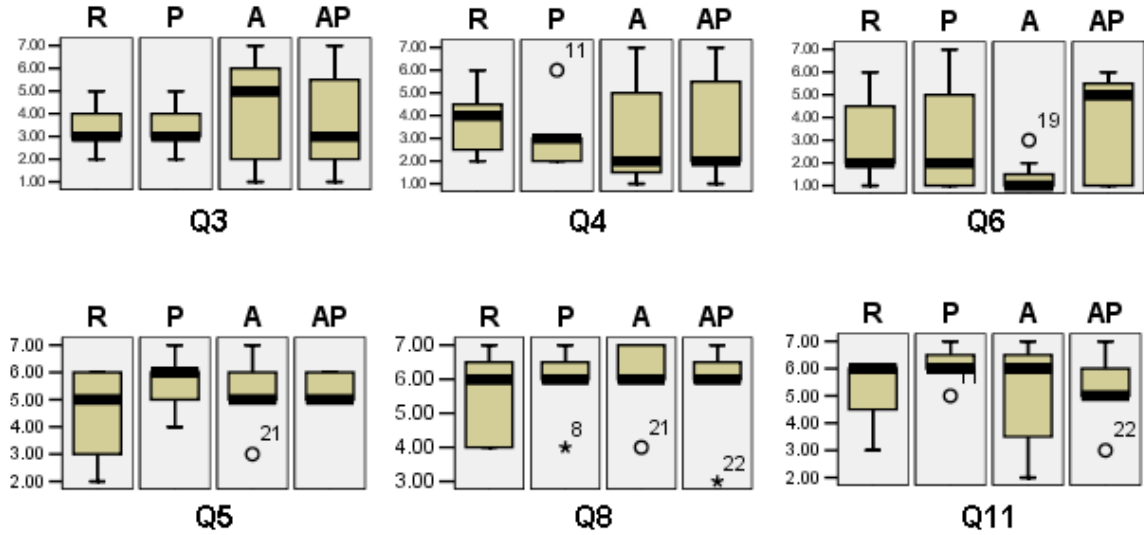


Figure 5.8: Training experience ratings - Q3: “the training session contained too much information and details”; Q4: “the training session was hard to follow and confusing”; Q6: “Information I needed to complete the task was missing in the training session”; Q5: “the training session was well pace”; Q8: “I feel comfortable using the Excel techniques showed in the training”; Q11: “overall, the training session was pleasant to watch”.

Training effectiveness by functionality

Subjects were asked to rate the effectiveness of the training they had for Excel functionality used in the tasks. Ratings were on a 5-point scale. A preliminary multivariate 2×2 ANOVA ($A \times P$) on the eight features detected a significant interaction effect only for training on “multiple sheets” functionality ($F_{1,24}=6.259$, $p=0.02$).

The ANOVA was followed by the more appropriate non-parametric Kruskal-Wallis tests on the ratings for “multiple sheets”. These showed that when $A=0$ (no augmentations) there was a significant effect of P resulting in a higher rating for effectiveness ($\chi^2(1) = 3.857$, $p = 0.05$, $\text{Med}(R)=3$, $\text{Avg}(R)=3.714$, $\text{SD}(R)=0.951$; $\text{Med}(P)=5$, $\text{Avg}(P)=4.714$, $\text{SD}(P)=0.756$). In contrast, when $A=1$ (with augmentation) there was no simple effect for P . No statistically significant effects for A were found. No significant effects for the other functionalities were found.

The results summary in Table 5.3 and graphs in Figure 5.9 show the same pattern, albeit without statistical significance, repeated for all eight functionalities. Participants in conditions P and A tended to give higher ratings than did participants in the control group (R).

Participants in condition AP showed more diversity between the different functionalities. While the training on conditional formatting, absolute references, lookup tables and the round function got relatively high ratings, ratings for the other functionalities spread across the entire rating gamut.

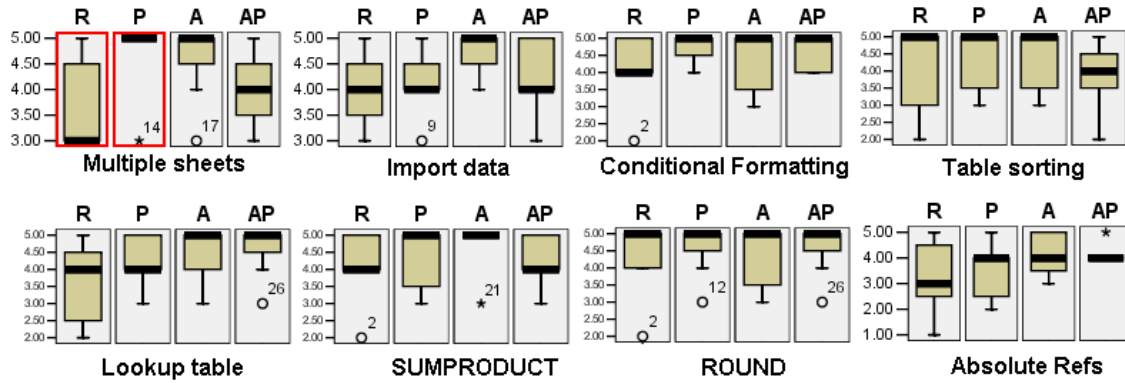


Figure 5.9: Rating of effectiveness of training per taught functionality. The only two statistically significant different groups are marked in red (“Multiple sheets” R and P).

	R	P	A	AP
Mean	3.889	4.181	4.329	4.187
Median	4.000	5.000	5.000	4.000
SD	1.123	0.872	0.806	0.756

Table 5.3: Overall functionality training ratings per condition. In general subjects from conditions P and A gave a higher rating than than did those in the control condition.

Visual enhancements ratings

Subjects were asked to rate the different augmentation and privacy features that were used in the training movies on a 5-point scale. Some participants rated features they saw only in the introduction movie and not in the training. These ratings were not counted in the analysis. Participants also had the option to mark N/A on features they did not notice in the training.

The 5-point scale was collapsed to 3 points (effective + very effective, neutral, disturbing and very disturbing). The graphs in Figure 5.10 summarize the results.

Augmentations

These features were rated by subjects in the A and AP conditions. We conducted t-tests that showed that there was no difference between these two groups and therefore all subjects with A=1 were analyzed together.

Rated most effective were the automated highlighting of cell changes (rated by 92.86% as effective), active dialog fields (rated by 91.67% as effective), and reporting on keyboard shortcuts (rated by 84.62% as effective).

Somewhat less effective was the highlighting of active selection context (rated by 61.54% as effective and by 38.46% as neutral).

Highlighting of mouse clicks was not considered to be effective (only 50% rated them as effective). This is somewhat surprising because similar highlighting is considered as attractive feature of commercial screen recording tools.

Menu augmentations received more diverse ratings with only 53.85% of the subjects rating extended menu highlighting as effective and 50.0% rating subtitles as effective. Few subjects considered these features disturbing (1 subject, 7.14% on subtitles and 2 subjects, 15.38% on menu highlighting). However, the findings from the performance analysis indicate that these features have some significant effect on menu selection times that outweigh the minor opinion about their disturbance.

Privacy filters

These features were rated by subjects in the P and AP conditions. We conducted t-tests that showed that there was no difference between these two groups and therefore all subjects in P=1 were analyzed together. Replacing dialogs with icons was also rated by subjects in condition A, again a one-way ANOVA test showed there was no significant difference in ratings between the A, P and AP groups with respect to rating this feature.

All privacy filters were classified as disturbing by some of the subjects, but no filter was categorically considered as disturbing (ranked so by more than 50% of the subjects). It is only natural that these filters be considered as somewhat disturbing because they hide information from the viewer.

Cell content blurring and concealing private worksheets received similar ratings. It is also not surprising that these features were considered to be disturbing by about 40% of the subject because they directly interfere with their ability to learn the task. However, it is surprising to see that a substantial fraction of the subjects (about 45%) were not bothered

by these privacy filters and that some other subjects (about 15%) thought the filters were effective.

More controversial were menu item blurring and replacing dialogs with icons. Menu item blurring was rated by 46.15% as effective and by 23.08% as disturbing, suggesting it is a viable filter. Replacing dialogs with icons had an almost the inverse pattern with 42.11% rating it as disturbing and 26.32% rating it as efficient.

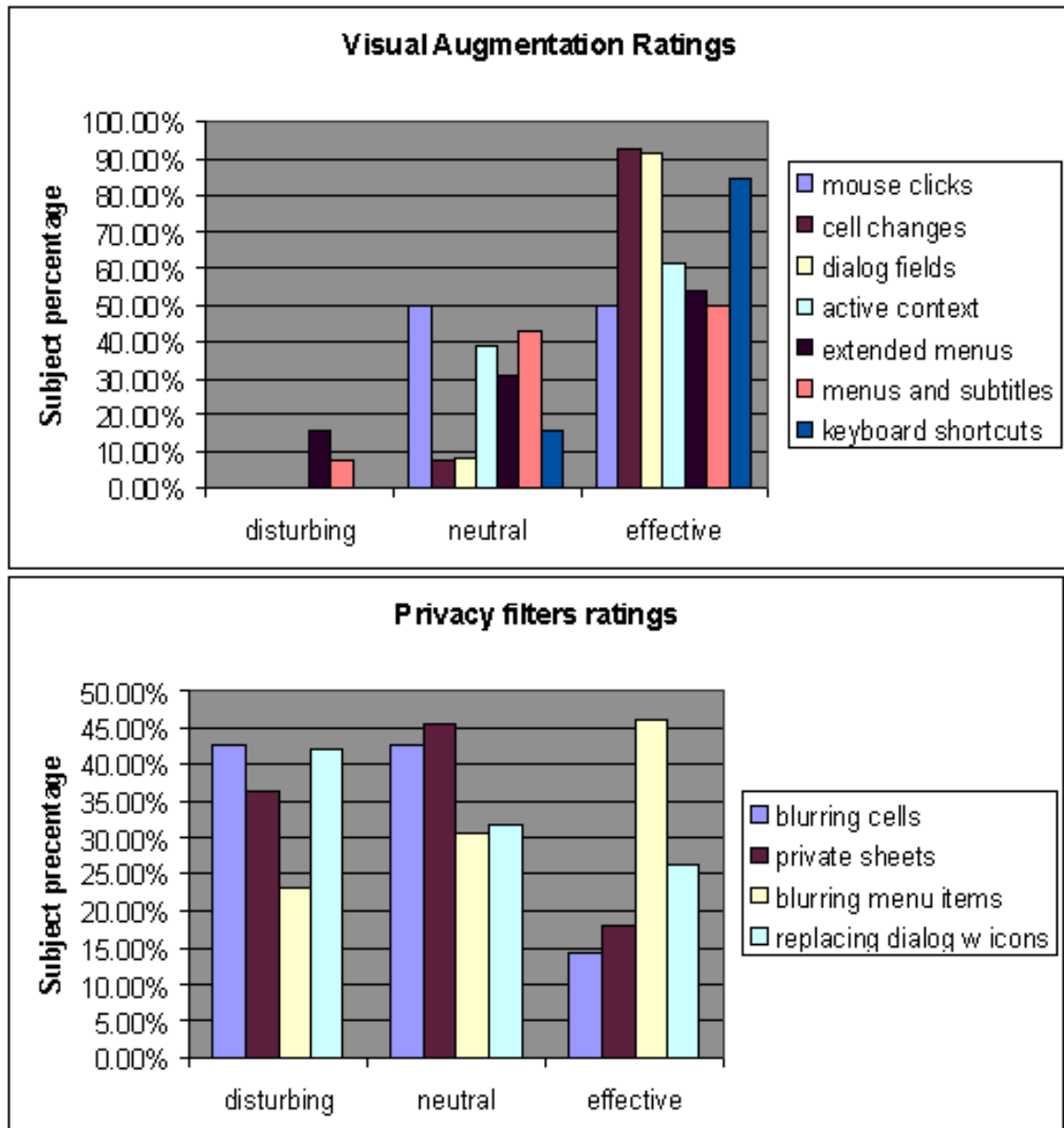


Figure 5.10: Ratings of augmentation features (top) and privacy filters (bottom).

Qualitative feedback

Participants also gave free form feedback in the questionnaires. We discuss some of the prominent remarks made by subjects:

Participants from all conditions suggested making the training clips shorter (“split videos into shorter sections”).

Several participants suggested watching the clips while doing the task or allowing a playback mode. Obviously allowing this can improve on the training and should probably be preferred in the real-world. However, allowing playback and a “do it as you go” as one participant suggested would probably diminish the differences between subjects and make the experiment analysis much harder.

Some participants in conditions R and P mentioned that the keyboard shortcuts used were hard to follow and reproduce - “some information like copy and past is not clear in the movie”, “Have some of the shortcuts in the movie as well (Ctrl+D)”. They would have also liked to see some of the other enhancements “the visual enhancements in the intro should be available in the training, they can help a lot”.

Some participants in these conditions also felt that the training was going too fast, “training was too fast. I could not remember how to use the techniques and had to guess”, “Some slowing down could be useful”.

Two participants in condition A found the training effective - “very efficient in teaching different Excel functionalities” and “It was a very good training session” and also one participant from condition R said - “Pretty good. To the point and concise”.

However, a few others (including several from condition AP) missed part of the visual enhancements, some of which went by too fast - “some substitutes were going too fast before catching my attention”, “payed more attention on finishing the task while ignoring the visual effect” and “the visual enhancements did not really enhance the experience. Some were unnecessary”.

5.4 Summary and conclusions

The experiment was conducted as part of the prototyping stage of the system development in order to identify gross patterns and relations between privacy and augmentation and to

identify promising directions for improvements.

5.4.1 Effects on performance

The experiment did not expose any global statistically significant differences in terms of speed and efficiency and therefore does not allow us to reject any of the four null hypotheses on these measures. The results of the experiment do suggest that introducing privacy filters does not necessarily impede performance. In fact, one secondary measure indicated that privacy filters may assist viewers - the reduced number of cell selection changes. This finding may be explained by the fact that Greeking cells in the training drew more attention to which cells were changed and which weren't.

In terms of accuracy, a statistically significant overall interaction $P \times A$ teaches us that H_{03} and H_{04} should be rejected.

Surprisingly, adding privacy filters improved accuracy when no visual augmentations were used in the training. Similarly, adding visual augmentations without privacy filters also improved accuracy (although significance was borderline). If both filter types are added there is no statistically significant effect on accuracy. Descriptive statistics suggest, however, that accuracy may be degraded in such cases.

With respect to H_4 , we cannot determine a global accuracy ordering of the four conditions, but can state based on the results that the four conditions are not equal as $f_{acc}(R) < f_{acc}(P)$ and $f_{acc}(R) < f_{acc}(A)$ ⁵. As for the other condition, we cannot determine its ordering based on the results, although descriptive statistics also suggest that for the particular set of filters used in the study the following may hold: $f_{acc}(AP) < f_{acc}(P) \approx f_{acc}(A)$.

The results do not allow us to reject H_{02} . Conversely, the overall and Task 2 simple main effects of privacy and the fact that none of the other tests showed a negative effect of privacy suggest that privacy filters do not reduce accuracy and may even improve it.

We could not globally reject H_{01} either, because there was no main effect for augmentation. However from the overall $TASK \times P \times A$ interaction detected we can deduce that adding visual filters is more effective for certain types of tasks but has little effect on other tasks. Moreover, the balance between privacy and augmentations should probably be tuned for the type of task.

Specifically, only Task 3 had a significant main effect of augmentation on accuracy.

⁵ $f_{acc}(\cdot)$ denotes the floating precision task grades that are believed to reflect accuracy.

Unlike Tasks 1 and 2 that were centered around one functionality type, the training for this task involved four or five different functionalities, some variations on previously taught commands and some new. This implies that visual augmentations are more effective when providing an overview of a complex task and less for a focused low-level discussion of the details of tasks. On the other hand, blurring filters were more useful in Task 2 that was based on detailed formula editing. A possible explanation is that blurring the cells forced viewers to focus on the formula bar and on the interaction mechanics, while the visual augmentations drew attention elsewhere. Although not designed for this purpose it seems that privacy filters can also serve as efficient augmentation means, proving that sometimes “less is more”.

Analysis of secondary performance measures indicated that providing augmentations for passive viewers significantly improves the ability to learn menu and keyboard commands. Also, subjects who had visual augmentations in their training relied less on the help mechanisms. These suggest that adding visual augmentations improves key performance aspects.

5.4.2 Balancing privacy and augmentations

Overall, the case for condition AP is particularly interesting. Our original conjecture was that this condition would be better than P, because some augmentations can help when data is masked. However, our experimental results suggest that using only privacy filters or only augmentations was better.

One possible explanation, examining Table 5.1, is that participants in condition AP happened to be less experienced with absolute referencing compared to the other conditions. This functionality was the basis for Task 2 and parts of Task 3; their relative lack of experience led to poor accuracy.

Another explanation is that using augmentations and privacy filters together resulted in a view that looked less like the real application interface that was used in the task. Alternatively there was simply too much information that was perceived as overwhelming by many viewers. This is supported to some extent by the questionnaire results. Subjects in condition A and AP expressed more concerns that there was too much information in the training (this was not statistically significant, however). Participants in the AP condition were particularly polarized in their opinions. Half of the AP subjects even thought that required information was missing in the training.

Clearly a better controlled study might provide a more accurate explanation. In any case an important conclusion is that one has to be careful about what visual effects are used, which effects can be combined, and how to balance the use of different filters.

5.4.3 Perceived utility

The questionnaire analysis supported the perceived utility of visual augmentations. Subjects who had visual augmentations in the training felt it was easier to complete the task and replicate the interactions from the training (statistically significant). Again, no significant effect for privacy was detected with respect to subject ratings of the training. This suggests that overall privacy filters are not perceived to degrade learning capabilities.

Supporting the global $A \times P$ interaction trend with respect to accuracy, subjects who had visual augmentations or privacy filters only thought the specific Excel functionality training was more effective than did subjects in the control group or subjects who had both types of filters.

In terms of rating the augmentations and filters used, it was not surprising that privacy filters were considered to be disturbing by subjects, although no single privacy filter was thought of as disturbing by more than half of the subjects. Blurring menu items was actually considered to be an effective filter by many subjects. These support the overall impression that privacy filters were not penalizing viewers.

As for augmentation effects, as expected, reporting on actions that otherwise have no visual indication or are hard to track got high ratings (keyboard shortcuts, cell changes and active context highlighting). More surprising was the relatively low rating of mouse click highlighting that is also used by commercial screen recorders, and the fact that few subjects thought menu selection highlighting was disturbing.

Together with the fact that some augmentations effects went by too fast for subjects this implies that the parameters and format of these effects should be further tuned and tested separately on subjects.

Chapter 6

Future Work and Conclusions

6.1 Future Work

The limitations discussed in Section 4.5 and the feedback collected on the prototype system point out prominent paths for future work and studies.

6.1.1 System improvements

We first discuss suggestions for improving the system.

Performance and network support

The current implementation of the system uses an inefficient timer-based copying of image buffers and cannot transfer these over the network. Implementing a variant of the RFB protocol can solve these limitations. More importantly, it will allow the system to generate more than one public view, which is useful in scenarios where meeting participants have varying degrees of privacy concerns or even for providing the presenter with custom hints on his view.

When considering the broadcast of shared application views over the network there are different possible models of where privacy filters and other window manipulations occur (the presenter's machine, the viewer's machine, or a trusted third party server). These models will each have different implications on performance, privacy and security. An interesting direction is mapping privacy and augmentation concerns in shared application viewing sessions to the possible architectures (taking into account privacy risk management concerns).

Improving the extraction of privacy risks

An important direction for future research is automating, simplifying and customizing the extractions of private elements or elements that need highlighting. Some initial techniques were introduced in the prototype but need more study: the “scripted hints”, mechanisms for specifying and applying policies, and a possible UI layer on top that will allow easy customization but still will require some effort from a presenter (although it is reasonable to assume that pre-cooked scripts for popular applications will be shared through public repositories, like the Mozilla extensions model, Section 4.1.3). Some basic automated heuristics for extracting private elements, states, and augmentation hints were also introduced (e.g. attempting to classify text in the document as sensitive or classify dialogs and fields by their name and context) but clearly more work is required. Even if automation is achieved, part of the problem as discussed in Chapter 2 is that privacy perception is subjective and varies between different presenters and viewers.

One possible direction for future research is applying user modelling and machine learning techniques to learn what elements are considered private by a presenter (and different audiences) and apply these to search the widget space and document model. Another direction is harnessing “programming by example” techniques. Perhaps most interesting are approaches that attempt to blend all of the techniques together, also known as mixed-initiative models. Previous work on such models, such as by Horvitz (1999) showed interesting results with respect to modelling application users to offer help in critical moments. Similar techniques may be able to address privacy risks as they arise.

Viewer control

More research is required on effective ways for viewer control and feedback. The prototype system does not handle any input from viewers (apart from a possible cursor control), especially with respect to manipulating the timeline of recorded interactions (playing back, slowing down, indexing and search) or on ways to control the views (viewers should be able to independently choose what augmentations they want or notify the presenter that they want them).

We did not focus on the sharing of multiple application views on a single screen. Particularly interesting are awareness applications and desktop monitoring solutions that have

privacy problems but can also benefit from indexing and augmentation techniques. Also interesting are wall displays or tabletop displays with high resolution where several people can work off the display simultaneously; they need to keep some information private as well as maintain mutual awareness.

6.1.2 Future studies

The results of the user study conducted indicate a number of areas in which we need to better understand the implications of the different filters as well as to improve their design.

Controlled study

In the study we chose to work with a “real-world” application and with tasks that were quite large in scope and represent actual tasks performed by users. However, it was hard to control subjects’ expertise with the application and particularly with specific features (such as absolute referencing).

The next version of the study should be based on a made-up system or a system that is unfamiliar to all subjects. Furthermore it will be beneficial to use smaller tasks or even focus on atomic operations (such as single menu selections or dialogs).

Individual feature tuning

The study used a fixed subset of augmentation features and privacy filters that were either all present or all missing. This scheme did not allow the results to be attributed to a particular feature. Moreover, qualitative feedback collected from several subjects indicates that some augmentation features, such as subtitles, went by too fast or were not salient enough and therefore were missed by them.

These call for an additional set of studies that will test different subsets of augmentation and privacy filters as well as individual filters. Another set of studies should focus on specific filters and attempt to find their optimal parameters (such as “decay” time, highlighting effect, location on screen, etc.).

Measuring utility for a presenter and other audience profiles

The user study focused, by choice, on the system effects on viewers in a training scenario. A different kind of user study is required for understanding how useful the system is for

presenters. We believe this requires a field study where data collected will indicate how presenters use the system, what parameters are more useful, what information is considered private and how they handle it. It is also important to understand how much effort presenters are willing to put into tagging private information as some parts of the system require.

As for viewers, there are other generalized presentation scenarios apart from training new users that can benefit from the system. For instance, explaining a report to viewers who are already familiar with the software tool that is being used. In this case other measures, such as comprehension and not task performance are more relevant. Task 3 in the study also provided an indication that when viewers are already familiar with the fine interaction details, having augmentations can have a stronger effect on performance. Clearly more studies with different task types and different viewer profiles are required.

6.2 Conclusions

We have introduced a unified solution for privacy concerns and verbosity control to assist a presenter and her audience in generalized presentation scenarios.

These concerns are not addressed by current single-user application sharing modes. Still, we constantly choose to “post” our desktop in public or share application views while knowing all too well that they are full of private and embarrassing information or that they contain too many irrelevant components and details. The regularity with which this happens is ample justification for tackling this problem.

Our design introduces role-driven views for each type of participant, balancing between the presenter’s privacy needs and the audience’s awareness needs. While such view disparity can be achieved through the use of collaboration-aware applications, in reality the majority of shared view sessions use off-the-shelf collaboration-unaware applications. Furthermore, most privacy concerns in such settings are not real security threats. The incentive to abandon familiar tools for more sophisticated ones is low.

Our system is based on applying image filters and spatial and timeline manipulations to bitmap representations of shared windows. The system’s framework is general and works with off-the-shelf applications, requiring a limited “semantic glue” layer introduced through an extensible plug-in architecture to monitor the visible information in an application and

drive these manipulations. The system allows additional intermediate sharing layers beyond the conventional screen, application or window layers of centralized application view sharing, expanding this part of the Zipper model (Dewan 1999).

Lessons learned from the prototype development

A prototype of the system was created and tested with several commercial applications. As part of our work we learned a few lessons about the feasibility of such a glue layer. The most prominent lesson is that useful information about an application's UI and window set for driving privacy and augmentation filters is already in place and it can be extracted through existing mechanisms from collaboration-unaware applications (Accessibility APIs being the most general and fruitful channel). We also learned that simple application-specific scripting can be used to parse the visual surface of an application and enrich the system's capabilities.

Our prototype demonstrates that applying generic image and window set filters based on the extracted information can satisfy useful privacy policies and provide an improved presentation experience. However, it was also learned that extended customization, policy and rule definition capabilities are required on top of the plug-ins. Partial implementations and potential directions for such extensions were introduced, namely using simple scripts and XML tables to customize manipulation rules and then blending these with policy and role specification languages that are normally used for access control.

It was important to verify the utility of the suggested manipulations and the system's framework before delving into richer policy specification and scripting techniques. Therefore a user study was conducted.

Lessons learned from the user study

The effects of using privacy and augmentation filters on viewers were tested in a training scenario. The results indicate that while privacy filters protect information that is important for the presenter, they do not interfere with viewer's ability to follow the training. In fact, privacy filters even improved some aspects of viewer learning and task performance (such as accuracy and some forms of efficiency), acting as augmentation and awareness features. It is interesting to compare these results with the use of blurring filters in video. These have been recently shown by Neustaedter et al. (2005) to be unsuitable for balancing privacy

and awareness. The difference is that the latter are not driven by “semantic glue”.

Augmentation manipulations and filters were effective for some task types (training semi-expert users on a wide set of functionalities) and not so much for other types (focused and detailed training of novice users). They were shown to be useful for eliciting interactions that lack proper feedthrough (keyboard shortcuts and menu selections).

These results indicate that the system can improve application view sharing sessions. It was also evident that some combinations of augmentation and privacy filters are not efficient or are even counter-productive. One conclusion is that a more accurate mapping of the different filter effects is required. Such a mapping should also be part of the policy and rule base that control the generation of shared views.

We believe that the results of the study and the potential utility of visual manipulations can inform the design of similar techniques in collaboration-aware tools as well as in asynchronous view sharing tools, such as screen recorders (in fact, the live output from the system was recorded as training movies for the study).

Contribution

Working with the system prototype and its use in the user study showed that the suggested approach can serve as a relatively simple alternative for enhancing a shared view of off-the-shelf applications, maintaining the key advantages of the popular bitmap-based sharing solutions - no code changes are required and viewers do not need a copy of the application.

The system improves the quality of generalized presentation sessions. It protects a presenter from exposing private information and elements, allowing her to work normally and comfortably. It assists viewers in maintaining a suitable level of awareness and in better understanding the presenter’s intentions.

In conclusion, this work has contributed to the emerging field of application view sharing in the following ways:

- We extended and developed a taxonomy of primitives for role-based view modification. Examined the key problems and their sources and identified different ways to solve them.
- We developed a model for role based view control that is largely application-independent. This model lies in a new spot in the application sharing architecture space. It is still

very close to collaboration-transparency solutions with their advantages, but extends existing information channels to parse an application's visual surface and window set.

- We designed and prototyped a system architecture to provide view control. Tested the system with various popular applications.
- We formally evaluated a set of augmentation and privacy filters using the system. The study provided important and somewhat surprising results on the utility of the different filters and their combinations. Namely, it pointed out that privacy filters may support viewers while also supporting the presenter.

Bibliography

- Abdel-Wahab, H. M. and M. Feit (1991). XTV: A framework for sharing X Window clients in remote synchronous collaboration. In *IEEE Conference on Communications Software: Communications for Distributed Applications & Systems*. ACM Press.
- Baecker, R. M. and I. S. Small (1990). Animation at the interface. In *The Art of Human-Computer Interface Design*, pp. 251–267. Addison-Wesley.
- Baudisch, P., E. Cutrell, and G. G. Robertson (2003). High-density cursor: A visualization technique that helps users keep track of fast-moving mouse cursors. In *Proceedings of INTERACT 2003*, pp. 236–243.
- Begole, J., M. B. Rosson, and C. A. Shaffer (1999). Flexible collaboration transparency: supporting worker independence in replicated application-sharing systems. *ACM Trans. Comput.-Hum. Interact.* 6(2), 95–132.
- Berry, L., L. Bartram, and K. S. Booth (2005). Role-based control of shared application views. In *Proceedings of the 17th annual ACM symposium on User interface software and technology (to appear)*. ACM Press.
- Bier, E. A., M. C. Stone, K. Pier, W. Buxton, and T. D. DeRose (1993). Toolglass and magic lenses: the see-through interface. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, New York, NY, USA, pp. 73–80. ACM Press.
- Blackwell, A. F., A. R. Jansen, and K. Marriott (2000). Restricted focus viewer: A tool for tracking visual attention. In *Diagrams '00: Proceedings of the First International Conference on Theory and Application of Diagrams*, London, UK, pp. 162–177. Springer-Verlag.
- Booth, K. S., B. D. Fisher, C. J. R. Lin, and R. Argue (2002). The “Mighty Mouse” multi-screen collaboration tool. In *UIST '02: Proceedings of the 15th annual ACM symposium on User interface software and technology*, New York, NY, USA, pp. 209–212. ACM Press.
- Boyle, M. and S. Greenberg (2005). The language of privacy: Learning from video media space analysis and design. *ACM Trans. Comput.-Hum. Interact.* 12(2), 328–370.
- Cheng, L.-T., S. L. Rohall, J. Patterson, S. Ross, and S. Hupfer (2004). Retrofitting collaboration into UIs with aspects. In *CSCW '04: Proceedings of the 2004 ACM conference on Computer supported cooperative work*, New York, NY, USA, pp. 25–28. ACM Press.
- Csinger, A., K. S. Booth, and D. Poole (1994). AI meets authoring: User models for intelligent multimedia. *Artif. Intell. Rev.* 8(5-6), 447–468.

- Dewan, P. (1999). Architectures for collaborative applications. In M. Beaudouin-Lafon (Ed.), *Computer Supported Co-operative Work*, Volume 7 of *Trends in Software*, pp. 169–193. John Wiley & Sons.
- Edwards, W. K. (1996). Policies and roles in collaborative applications. In *CSCW '96: Proceedings of the 1996 ACM conference on Computer supported cooperative work*, New York, NY, USA, pp. 11–20. ACM Press.
- Edwards, W. K., S. E. Hudson, J. Marinacci, R. Rodenstein, T. Rodriguez, and I. Smith (1997). Systematic output modification in a 2D user interface toolkit. In *UIST '97: Proceedings of the 10th annual ACM symposium on User interface software and technology*, New York, NY, USA, pp. 151–158. ACM Press.
- Greenberg, S., M. Boyle, and J. Laberg (1999). PDAs and shared public displays - making personal information public and public information personal. *Personal Technologies* 3(1), 54–64.
- Greenberg, S. and M. Roseman (1999). Groupware toolkits for synchronous work. In M. Beaudouin-Lafon (Ed.), *Computer Supported Co-operative Work*, Volume 7 of *Trends in Software*, pp. 135–168. John Wiley & Sons.
- Greenberg, S. and M. Rounding (2001). The notification collage: posting information to public and personal displays. In *CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 514–521. ACM Press.
- Grudin, J. (1988). Why CSCW applications fail: problems in the design and evaluation of organization of organizational interfaces. In *CSCW '88: Proceedings of the 1988 ACM conference on Computer-supported cooperative work*, New York, NY, USA, pp. 85–93. ACM Press.
- Grudin, J. (1994). Computer-supported cooperative work: History and focus. *Computer* 27(5), 19–26.
- Gutwin, C. and S. Greenberg (1998a). 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*, New York, NY, USA, pp. 207–216. ACM Press.
- Gutwin, C. and S. Greenberg (1998b). Effects of awareness support on groupware usability. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 511–518. ACM Press/Addison-Wesley Publishing Co.
- Hawkey, K. and K. M. Inkpen (2005). Privacy gradients: exploring ways to manage incidental information during co-located collaboration. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, New York, NY, USA, pp. 1431–1434. ACM Press.
- Hexel, R., C. Johnson, B. Kummerfeld, and A. Quigley (2004). "PowerPoint to the people": suiting the word to the audience. In *CRPIT '04: Proceedings of the fifth conference on Australasian user interface*, Darlinghurst, Australia, Australia, pp. 49–56. Australian Computer Society, Inc.

- Hong, J. I., J. D. Ng, S. Lederer, and J. A. Landay (2004). Privacy risk models for designing privacy-sensitive ubiquitous computing systems. In *DIS '04: Proceedings of the 2004 conference on Designing interactive systems*, New York, NY, USA, pp. 91–100. ACM Press.
- Horvitz, E. (1999). Principles of mixed-initiative user interfaces. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, New York, NY, USA, pp. 159–166. ACM Press.
- Hutchings, D. R. and J. Stasko (2004). Revisiting display space management: understanding current practice to inform next-generation design. In *GI '04: Proceedings of the 2004 conference on Graphics interface*, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, pp. 127–134. Canadian Human-Computer Communications Society.
- Hutchings, D. R. and J. Stasko (2005). Mudibo: multiple dialog boxes for multiple monitors. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, New York, NY, USA, pp. 1471–1474. ACM Press.
- Johanson, B., G. Hutchins, T. Winograd, and M. Stone (2002). PointRight: experience with flexible input redirection in interactive workspaces. In *UIST '02: Proceedings of the 15th annual ACM symposium on User interface software and technology*, New York, NY, USA, pp. 227–234. ACM Press.
- Lau, T., O. Etzioni, and D. S. Weld (1999). Privacy interfaces for information management. *Commun. ACM* 42(10), 88–94.
- Lederer, S., J. Mankoff, and A. Dey (2003). Towards a deconstruction of the privacy space. Technical Report IRB-TR-03-037, Intel-Research Berkley.
- Li, D. and R. Li (2002). Transparent sharing and interoperation of heterogeneous single-user applications. In *CSCW '02: Proceedings of the 2002 ACM conference on Computer supported cooperative work*, New York, NY, USA, pp. 246–255. ACM Press.
- Li, S. F., M. Spiteri, J. Bates, and A. Hopper (2000). Capturing and indexing computer-based activities with virtual network computing. In *SAC '00: Proceedings of the 2000 ACM symposium on Applied computing*, New York, NY, USA, pp. 601–603. ACM Press.
- Lok, S., S. K. Feiner, W. M. Chiong, and Y. J. Hirsch (2002). A graphical user interface toolkit approach to thin-client computing. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, New York, NY, USA, pp. 718–725. ACM Press.
- Mackinlay, J. (1986). Automating the design of graphical presentations of relational information. *ACM Trans. Graph.* 5(2), 110–141.
- McGrenere, J. L. (2002). *The design and evaluation of multiple interfaces: a solution for complex software*. Ph. D. thesis. Adviser-Ronald Baecker and Adviser-Kellogg Booth.
- Myers, B. A., C. H. Peck, J. Nichols, D. Kong, and R. Miller (2001). Interacting at a distance using semantic snarfing. In *UBICOMP '01: Proceedings of the 3rd international conference on Ubiquitous Computing*, London, UK, pp. 305–314. Springer-Verlag.

- Neustaedter, C., S. Greenberg, and B. Michael (2005). Blur filtration fails to preserve privacy for home-based video conferencing. *ACM Transactions on Computer-Human Interaction (to appear)*.
- Norman, D. A. (1988). *The Design of Everyday Things*. New York: Doubleday.
- Olsen, D. R., D. Boyarski, T. Verratti, M. Phelps, J. L. Moffett, and E. L. Lo (1998). Generalized pointing: enabling multiagent interaction. In *CHI '98: Proceedings of the SIGCHI conference on Human factors in computing systems*, New York, NY, USA, pp. 526–533. ACM Press/Addison-Wesley Publishing Co.
- Olsen, D. R., S. E. Hudson, T. Verratti, J. M. Heiner, and M. Phelps (1999). Implementing interface attachments based on surface representations. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 191–198. ACM Press.
- Olsen, D. R., T. Tauber, and J. A. Fails (2004). Screencrayons: annotating anything. In *UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology*, New York, NY, USA, pp. 165–174. ACM Press.
- Palen, L. and P. Dourish (2003). Unpacking “privacy” for a networked world. In *Proceedings of the conference on Human factors in computing systems*, pp. 129–136. ACM Press.
- Po, B. A., B. D. Fisher, and K. S. Booth (2005). Comparing cursor orientations for mouse, pointer, and pen interaction. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, New York, NY, USA, pp. 291–300. ACM Press.
- Reeves, S., S. Benford, C. O'Malley, and M. Fraser (2005). Designing the spectator experience. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, New York, NY, USA, pp. 741–750. ACM Press.
- Rekimoto, J. and M. Saitoh (1999). Augmented surfaces: a spatially continuous work space for hybrid computing environments. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, New York, NY, USA, pp. 378–385. ACM Press.
- Richardson, T., Q. Stafford-Fraser, K. R. Wood, and A. Hopper (1998). Virtual network computing. *Internet Computing, IEEE* 2(1), 79–109.
- Roseman, M. and S. Greenberg (1996). Building real-time groupware with groupkit, a groupware toolkit. *ACM Trans. Comput.-Hum. Interact.* 3(1), 66–106.
- Sakairi, T., M. Shinozaki, and M. Kobayashi (1998). CollaborationFramework: A toolkit for sharing existing single-user applications without modification. In *APCHI '98: Proceedings of the Third Asian Pacific Computer and Human Interaction*, pp. 183. IEEE Computer Society.
- Shoemaker, G. B. D. and K. M. Inkpen (2001). Single display privacyware: augmenting public displays with private information. In *CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems*, New York, NY, USA, pp. 522–529. ACM Press.
- Stefik, M., D. G. Bobrow, G. Foster, S. Lanning, and D. Tatar (1987). WYSIWIS revised: early experiences with multiuser interfaces. *ACM Trans. Inf. Syst.* 5(2), 147–167.

- Stoakley, R., M. J. Conway, and R. Pausch (1995). Virtual reality on a WIM: interactive worlds in miniature. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, New York, NY, USA, pp. 265–272. ACM Press/Addison-Wesley Publishing Co.
- Tan, D. S. and M. Czerwinski (2003). Information voyeurism: social impact of physically large displays on information privacy. In *CHI '03 extended abstracts on Human factors in computing systems*, pp. 748–749. ACM Press.
- Tan, D. S., B. Meyers, and M. Czerwinski (2004). WinCuts: manipulating arbitrary window regions for more effective use of screen space. In *CHI '04: CHI '04 extended abstracts on Human factors in computing systems*, New York, NY, USA, pp. 1525–1528. ACM Press.
- Terry, M. and E. D. Mynatt (2002). Side Views: persistent, on-demand previews for open-ended tasks. In *UIST '02: Proceedings of the 15th annual ACM symposium on User interface software and technology*, New York, NY, USA, pp. 71–80. ACM Press.
- Thomas, B. H. and P. Calder (2001). Applying cartoon animation techniques to graphical user interfaces. *ACM Transactions on Computer-Human Interactions* 8(3), 198–222.
- Tolone, W., G.-J. Ahn, T. Pai, and S.-P. Hong (2005). Access control in collaborative systems. *ACM Comput. Surv.* 37(1), 29–41.
- Xia, S., D. Sun, C. Sun, D. Chen, and H. Shen (2004). Leveraging single-user applications for multi-user collaboration: the CoWord approach. In *CSCW '04: Proceedings of the 2004 ACM conference on Computer supported cooperative work*, pp. 162–171. ACM Press.
- Yerazunis, W. and M. Carbone (2001). Privacy-enhanced displays by time-masking images. Technical Report TR2002-011, Mitsubishi Electric Research Laboratories.

Appendix A

Questionnaire

The following questionnaire was administered to all subjects in the user study. The results were analyzed and are reported in Chapter 5 of this thesis.

The user study was conducted under the auspices of ethics certificate B03-0151 (amended) issued by the Behavioral Ethics Research Board of the University of British Columbia.



Collaborative Visualization and Interaction in Ubiquitous Computing Environments Study Questionnaire Form

Instructions

Please try to respond to all of the items listed below. For those items that are not applicable, specify N/A. If you have any comments, please be sure to write them down in Part 4.

Part 1: Past Computer Experience and Excel Experience

1. How often do you use a computer?

☐ Never ☐ Once a month ☐ Once a week ☐ Every 2-3 days ☐ Every day

2. How often do you use Excel ? (if using a different spreadsheet application, specify which: _____)

☐ Never ☐ Once a month ☐ Once a week ☐ Every 2-3 days ☐ Every day

3. How would you rate your overall expertise level with Excel (or a different spreadsheet you use) ?

☐ None ☐ Basic ☐ Intermediate ☐ Expert

4. For each Excel functionality listed below: a. How familiar were you with it prior to the training session ?

b. How effective was the training session in teaching you how to use the functionality in the task?

Functionality	Prior Familiarity			Training Session Effectiveness							N/A
	Not familiar	Basic	Expert		1	2	3	4	5		
Multiple sheets	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Highly Ineffective	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Highly Effective	<input type="checkbox"/>
Importing external data	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Highly Ineffective	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Highly Effective	<input type="checkbox"/>
Lookup table	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Highly Ineffective	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Highly Effective	<input type="checkbox"/>
Conditional formatting	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Highly Ineffective	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Highly Effective	<input type="checkbox"/>
SUMPRODUCT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Highly Ineffective	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Highly Effective	<input type="checkbox"/>
Absolute references (\$)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Highly Ineffective	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Highly Effective	<input type="checkbox"/>
Table Sorting	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Highly Ineffective	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Highly Effective	<input type="checkbox"/>

Part 2: Training session and Task Experience

			1	2	3	4	5	6	7		N/A
1.	The training session helped me perform the task better	strongly disagree	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	strongly agree	<input type="checkbox"/>
2.	The training session was effective	strongly disagree	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	strongly agree	<input type="checkbox"/>
3.	The training session contained too much information and details	strongly disagree	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	strongly agree	<input type="checkbox"/>
			1	2	3	4	5	6	7		N/A

4.	The training session was hard to follow and confusing	strongly disagree	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	strongly agree	<input type="checkbox"/>
5.	The training session was well paced	strongly disagree	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	strongly agree	<input type="checkbox"/>
6.	Information I needed to complete the task was missing in the training session	strongly disagree	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	strongly agree	<input type="checkbox"/>
7.	Overall, it was easy to complete the task	strongly disagree	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	strongly agree	<input type="checkbox"/>
8.	I feel comfortable using the Excel techniques introduced in the training session	strongly disagree	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	strongly agree	<input type="checkbox"/>
9.	I easily understood the grade report structure	strongly disagree	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	strongly agree	<input type="checkbox"/>
10.	It was easy to replicate the presenter's actions	strongly disagree	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	strongly agree	<input type="checkbox"/>
11.	Overall, the training session was pleasant to watch	strongly disagree	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	strongly agree	<input type="checkbox"/>

Part 3: Visual Enhancements

How effective or disturbing were the following visual enhancements throughout the training session ? (specify N/A if you did not notice a particular visual enhancement)

		1	2	3 = Neutral	4	5	N/A
1. Visual indication of mouse clicks	disturbing	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	effective	<input type="checkbox"/>
2. Highlighting of cell changes	disturbing	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	effective	<input type="checkbox"/>
3. Highlighting of active dialog fields	disturbing	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	effective	<input type="checkbox"/>
4. Highlighting or dimming of active context on spreadsheet (table, row or column)	disturbing	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	effective	<input type="checkbox"/>
5. Extended highlighting and blinking of menu selections	disturbing	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	effective	<input type="checkbox"/>
6. Replacing dialogs (e.g. File dialog, wizard) with an iconic representation	disturbing	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	effective	<input type="checkbox"/>
7. Reporting menu selections as subtitles	disturbing	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	effective	<input type="checkbox"/>
8. Reporting keyboard shortcuts as subtitles	disturbing	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	effective	<input type="checkbox"/>
9. Blurring/Masking private data in cells	disturbing	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	effective	<input type="checkbox"/>
10. Iconic indicators on private sheets	disturbing	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	effective	<input type="checkbox"/>
11. Blurring menu items / dialog items	disturbing	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	effective	<input type="checkbox"/>

Part 4: Comments and Suggestions (next page)

Please provide any other comments or suggestions that you might have.

Appendix B

Task Descriptions

Each participant completed three subtasks based on a training movie and task notes sheets that were handed out (section B.2). The fourth task was a combination of all three subtasks and no training movie was shown.

The Excel sheets filled for each subtask and final task were graded by a single marker based on a fixed marking scheme that will also be presented in this appendix. The tasks identical for all subjects and were mostly mechanical and so is the marking scheme.

Although in the general case, two independent markers should define the grade on a task (assuring inter-grader reliability), we believe that in this case the grades are objective enough.

The marking scheme tries to give proper weights to each functionality type and capture common errors or actions that subjects forgot from the training sessions as these are indicators on the quality of the training. Also, subjects who came up with a solution or partial solution that worked but diverged from the method shown in the training were penalized accordingly (if it was close enough they got half points and no points otherwise).

For the statistical data analysis, described in chapter 5, grades were later converted to a floating scale in the range [0-1].

B.1 Marking Scheme

Task 1

Total: 10 points

Filling cover page

- 1 pt - filling in course number
- 1 pt - filling in course name and instructor name
- 1 pt - filling in user id and e-mail
- 1 pt - creating a new worksheet “students”
- 1 pt - creating a new worksheet “grades”

Importing data

- 1 pt - Importing students list

- 1 pt - Importing grades
- 1 pt - Fixing missing grades (Find and Replace)

Preparing cover page for entering students

- 1 pt - Copying student numbers from grades list to cover page

Task 2

Total: 10 points

Computing student programs and names

- 1 pt - correct lookup value
- 1 pt - correct search table
- 1 pt - correct column and using FALSE on range lookup
- 1 pt - program computed for all students (use of \$ on the table ref)
- 2 pt - lookup formula for students name (1 pt if correct + 1pt if lookup value uses \$)

Computing grades

- 1 pt - Copy grades header row from “grades” sheet
- 3 pts - Compute grades for students (1pt first grade for first student, 1pt first line computed correctly with \$ on lookup value, 1pt all lines computed correctly with \$ on table ref)

Task 3

Total: 15 points

Computing final grades

- 1 pt - enter grade component weights
- 4 pts - SUMPRODUCT (1 pt for vector of grades, 1 pt for vector of weights, 1 pt for \$ on weights, 1 pt for using ROUND)

Computing letter grades

- 1 pt - HLOOKUP and correct lookup value
- 1 pt - correct table
- 1 pt - using TRUE on range lookup
- 1 pt - \$ on table

Finalizing report

- 1 pt - conditional formatting
- 1 pt - sorting
- 1 pt - using AVERAGE and ROUND for each grade component
- 1 pt - using STDEV and ROUND for each grade component
- 1 pt - linking final grade average and standard deviation to the cover page

Task 4

Total: 35 points

Part 1

10 points

- 1 pt - filling course number of cover page
- 1 pt - filling course name and instructor name
- 1 pt - filling name and e-mail
- 1 pt - filling date
- 2 pt - creating “students” sheet and importing students list
- 2 pt - creating “grades” sheet and importing grades list
- 1 pt - fixing missing grades
- 1 pt - copying students numbers from grades list to cover page

Part 2

10 points


- 1.25 pt - use correct lookup value for computing program of first student
- 1.25 pt - using correct table in lookup
- 1.25 pt - correct column
- 1.25 pt - \$ on table ref
- 1.25 pt - \$ on table lookup value
- 1.25 pt - use lookup for first grade of first student
- 1.25 pt - compute grades of first line correctly (\$ on lookup value)
- 1.25 pt - compute grades for all lines (\$ on table)


Part 3


15 points

- 1 pt - enter grade weights
- 4 pt - SUMPRODUCT (same as for task 3)
- 4 pt - letter grades (same as for task 3)
- 2 pt - conditional formatting
- 1 pt - sorting
- 2 pt - ROUNDED average and standard deviation for all grade components
- 1 pt - linking final grade average and standard deviation to the cover page

B.2 Task descriptions

	Task Instructions
Scenario	
<p>Hi,</p> <p>You are a new group assistant in the Department of Computer Science, UBC. Your boss asked you to complete a grade report for the course: CPSC 534, "Computational Computations". All you have is a partial report template in Excel, a text file with the raw grades and a text file with students' information.</p> <p>Another group assistant, Joe, has agreed to show you how to fill the report. Joe is logging in from home and uses NetMeeting and VNC to give you a tutorial and demonstrate good practices for filling in such a report. He will use his own student data (from the CPSC 522, HCI course) and a similar report.</p> <p>Later you will be asked to complete your grade report using the methods Joe showed you.</p>	
Experiment Procedure	
<p>The expected duration of the experiment is one hour. You will be going through the following parts:</p> <ol style="list-style-type: none">1. Fill and sign consent forms2. Watch a short explanation from Joe on his tutorial materials3. Watch Joe's mini-tutorials4. Complete the report after each mini-tutorial (you will be provided with notes from Joe's tutorial)5. Complete a short questionnaire6. Debriefing <p>For the movie parts you will be asked to put on headphones.</p> <p>- If you have any questions at this point please ask the experimenter.</p>	

	notes Task 1
	<ol style="list-style-type: none">1. The Excel document has already been opened for you (Report522.xls)2. At any point you can use Excel's help (F1)3. Fill in the course information part of the cover page. The course is 5224. use the userid assigned to you as your name and e-mail5. Insert a new worksheet named "students" and another worksheet named "grades"6. Import c:\report\students_522.txt and c:\report\grades_522.txt into the new sheets (respectively).7. Fix missing grades (find all #’s and replace with zeros)8. Copy student numbers from the grades list into the report9. Save the report10. Click the "Stop" button

	notes Task 2
	<ol style="list-style-type: none">1. The Excel document has already been opened for you (Report522.xls)2. At any point you can use Excel's help (F1)3. Compute the “program” for the first student on the report using a lookup formula4. Get the “program” computed for all students5. Compute the names for all students in the report6. Copy grade table header row into the report (override existing values)7. use a lookup formula to get the grades for all students computed8. Save the report9. Click the “Stop” button

**notes****Task 3**

1. The Excel document has already been opened for you (Report522.xls)
2. At any point you can Excel's help (F1)
3. Enter the weights for the grade components:

A1	A2	M	FE
10%	10%	30%	50%

4. Compute the **final grade** for each student, using a **SUMPRODUCT** formula
5. Make sure that the computed final grades are **rounded**.
6. Compute a **letter-grade** for each final grade, using the **conversion table** and a **lookup** formula
7. Add **conditional formatting** on the grades so that 0 grades appear with a **red background**
8. **Sort** the report table by **program** and then **name**
9. Compute **rounded average** and **standard deviation** (STDEV) for each grade component
10. **Link** the **average** and **standard deviation** of the final grades to the **cover page**
11. **Save** the report
12. Click the "**Stop**" button

**notes****Task 4**

1. The Excel document has already been opened for you (Report534.xls)
2. At any point you can Excel's help (F1)
3. Fill in the cover page. The course this time is **534**
4. use the userid assigned to you as your **name** and **e-mail**
5. import c:\report\students_534.txt and c:\report\grades_534.txt into new sheets ("students" and "grades")
6. Fix missing grades (# -> 0)
7. Copy student numbers from grades list into the report
8. Compute the "**program**", **first** and **last name** for all students on the report sheet using a **lookup** formula
9. Compute the **grades** for each student using a lookup formula
10. Enter the weights for the grade components:

A1	A2	A3	ME	FE
10%	10%	10%	25%	45%

11. Compute the **rounded final grade** for each student
12. Compute **letter grades** from each final grade
13. Add **conditional formatting** on the grades so that **all 0 grades** appear with a **red background**
14. Add another **condition** so that **all grades lower than 60** appear with a **yellow background**
15. **Sort** the report by **program**, then **last name**
16. Compute **rounded average** and **standard deviation** for each assignment
17. **Link** the **average** and **standard deviation** of the final grades to the **cover page**
18. **Save the report**
19. Click the "**Stop**" button