

Cite as:

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

THE UNIVERSITY OF CALGARY

Change Awareness in 2D Graphical Workspaces

by

James Roger Tam

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

CALGARY, ALBERTA

February, 2002

©James Roger Tam 2002

UNIVERSITY OF CALGARY
FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled "Change Awareness in 2D Graphical Workspaces" submitted by James Roger Tam in partial fulfillment of the requirements for the degree of Master of Science.

Supervisor, Saul Greenberg

Department of Computer Science

Co-Supervisor, Frank Maurer

Department of Computer Science

Sheelagh Carpendale,

Department of Computer Science

Armin Eberlein,

Department of Electrical & Computer Engineering

Date

Abstract

Change Awareness is the ability to track the changes of others in a collaborative project. In this thesis, I develop a framework for understanding how change awareness can be applied to 2D graphical applications. This framework articulates basic informational elements needed by a person to track changes in these systems. Additionally, I discuss the fundamental concepts for displaying these informational elements. These include the crucial display dimensions (placement, presentation and persistence), methods for reducing the time and effort to acquire and understand the information, mechanisms for displaying the elements, and filtering changes to avoid overload. Finally, I demonstrate the value of both the framework and the display concepts by showing how they can be used to critique a change awareness tool implemented as a part of my investigations. The critique not only pinpointed the weaknesses of the tool but how it could be redesigned to better support change awareness.

Publications from this thesis

Material, ideas and figures from Chapter 5 of this thesis have appeared previously in the following technical report:

Tam, J., McCaffrey, L., Maurer, F. and Greenberg, S. (2000) *Change Awareness in Software Engineering Using Two Dimensional Graphical Design and Development Tools*. Technical Report 2000-670-22, Department of Computer Science, University of Calgary (Calgary, Canada),
<http://www.cpsc.ucalgary.ca/grouplab/papers>

Acknowledgements

I would like to take this opportunity to thank all of the wonderful people in the Computer Science department who made me feel like I truly belonged. I would especially like to express my gratitude to the following people who were especially helpful to my research and kind to me even when I was lost in ‘awareness’ land.

To Saul for all your patience and understanding while I explored different paths even when they didn’t always match your own, your guidance whenever I got mired in detail, and support well above and beyond that required by a supervisor.

Frank - thanks for putting up with me while I decided upon the path that I wanted my research to take, and being the first one to offer me an opportunity to further my education in graduate studies.

Thank you to all my friends in Grouplab: Mike Boyle who not only shared his technical genius with us but also acted as the informal ‘big brother’ for everyone. Shaun who was always there behind me with either a helping hand or a smile of encouragement. Mike Rounding whom I always enjoyed talking with and could cheer me up by just being himself. Ana for her kindness, and providing us all the benefit of her knowledge and experience. John, the one who gave me so much good advice on more topics and on more occasions than I could keep track of. Also this is to my friends Chester, Carmen, Ed and Susannah. I will always fondly look back on my times in Grouplab.

My gratitude to Sheelagh who provided me with valuable advice and direction on the “artsy” parts of my thesis even though she was never required to. You helped open my eyes to the creative side of my research.

To my good friend Andy who kept my computer working even through the harshest of conditions.

Finally, I would like to thank Carl Gutwin for creating such an awesome Ph.D. thesis, which formed the basis for so much of my own work.

Dedication

This is for my parents, Tommy and Susie Tam. Thank you for having the foresight to travel to a wonderful country where I was lucky enough to be able to further my knowledge and experience the wonder of graduate school. I've always admired your strength and ability to endure: tough economic times, a world war and even all the goofy things I did. I hope that I can someday make you as proud of me as I am of the two of you.

Table of Contents

Approval Page.....	ii
Abstract.....	iii
Publications from this thesis.....	iv
Acknowledgements.....	v
Dedication.....	vii
Table of Contents.....	viii
List of Tables.....	xii
List of Figures.....	xiii
Chapter 1 Introducing change awareness.....	1
1.1 Definition of change awareness and the need for it in group work.....	1
1.2 Research context.....	4
1.3 Research objectives.....	5
1.4 Outline of this thesis.....	6
Chapter 2 Previous research into change awareness.....	8
2.1 Version control and configuration management systems.....	8
2.1.1 Manually tracking changes through backups.....	8
2.1.2 Version control systems.....	9
2.1.3 Configuration management systems.....	10
2.1.4 Limitations of version control and configuration management systems	
.....	11
2.2 Displaying change in text-based systems.....	14
2.2.1 Text based differencing.....	14
2.2.2 Extensions to the differencing algorithms: presenting changes at the	
document level.....	16
2.2.3 Presenting an overview of changes.....	21

2.2.4 Existing portrayals of changes in text-based systems.....	27
2.3 Change awareness in existing two dimensional graphical systems.....	27
2.4 Summary.....	32
 Chapter 3 Foundations of change awareness.....	34
3.1 Components of awareness.....	35
3.2 Gutwin's notion of workspace awareness.....	36
3.2.1 Workspace awareness in the present.....	38
3.2.2 Workspace awareness in the past.....	39
3.3 Informational elements for change awareness.....	41
3.3.1 Where?.....	43
3.3.2 Who?.....	45
3.3.3 What?.....	46
3.3.4 How?.....	47
3.3.5 When?.....	49
3.3.6 Why?.....	50
3.3.7 Discussion.....	51
3.4 Summarizing the components of awareness.....	52
 Chapter 4 Display of awareness information.....	56
4.1 Dimensions of change display.....	57
4.2 Reducing interpretation and acquisition costs.....	60
4.2.1 Reducing acquisition and interpretation costs.....	61
4.2.2 Representing information from the three workspace perspectives to reduce acquisition and interpretation costs.....	64
4.2.3 Reducing acquisition costs by displaying changes outside the workspace	65
4.3 Displaying change awareness information.....	68

4.3.1 Bertin's visual variables and the display of change awareness information.....	69
4.3.2 Other display mechanisms: storyboarding and text.....	79
4.4 Filtering the display.....	81
4.4.1 Types of filters.....	82
4.4.2 Filtering strategies.....	83
4.5 Summarizing the important concepts in the representation of change awareness information.....	88
 Chapter 5 Investigating change display mechanisms.....	90
5.1 Methodology.....	90
5.1.1 Test participants.....	91
5.1.2 Test materials.....	91
5.1.3 Test procedure.....	92
5.2 The change display mechanisms.....	93
5.3 Results and discussion.....	98
5.3.1 Findings about the display mechanisms.....	98
5.3.2 Findings about the display dimensions.....	100
5.3.3 General findings.....	101
5.3.4 Limitations of the study.....	101
5.4 Conclusions.....	102
 Chapter 6 Appraising the framework and display concepts via a sample change awareness implementation.....	105
6.1 Introduction to the base system and a usage scenario for PastDraw.....	106
6.2 Analyzing PastDraw's change tracking.....	112
6.3 Analyzing PastDraw's display of changes and suggested improvements.....	117
6.3.1 Critiquing PastDraw's display mechanisms.....	117

6.3.2. Classifying PastDraw's display mechanisms according to the display dimensions.....	123
6.3.3 Analyzing PastDraw's filtering mechanisms.....	127
6.4 Overall analysis and redesign of PastDraw.....	128
6.5 Discussion of the critique.....	132
 Chapter 7 Conclusion.....	135
7.1 Research objectives and contributions.....	135
7.2 Future work.....	137
 References.....	139
 Appendix A: Usability study materials.....	146
A.1 Per-participant procedures.....	146
A.2 Consent form.....	148
A.3 Pretest questionnaire.....	149
A.4 Post-Scenario Questions.....	150
A.5 Post-Test Questions.....	152

List of Tables

Table 3.1: Elements of WA relating to the present from Gutwin (1997).....	38
Table 3.2: Elements of WA relating to the past from Gutwin (1997).....	39
Table 3.3: Informational elements and specific workspace questions related to the ‘where’ aspect of change awareness.....	44
Table 3.4: Informational elements and specific workspace questions related to the ‘who’ aspect of change awareness.....	46
Table 3.5: Informational elements and specific workspace questions related to the ‘what’ aspect of change awareness.....	47
Table 3.6: Informational elements and specific workspace questions related to the ‘how’ aspect of change awareness.....	48
Table 3.7: Informational elements and specific workspace questions related to the ‘when’ aspect of change awareness.....	49
Table 3.8: Informational elements and specific workspace questions related to the ‘why’ aspect of change awareness.....	50
Table 3.9: Summary of all the change-related categories of questions, informational elements associated with those categories and examples of specific questions that may be asked.....	54
Table 4.1: Summarizing the effectiveness of different visual variables for different visual interpretation tasks, adapted from Bertin (1967) and Carpendale (2001).....	71
Table 5.1: Classification of the change display mechanisms employed.....	94
Table 5.2: Summary of the preferences of test participants.....	98
Table 6.1: Summary of PastDraw's change tracking.....	113
Table 6.2: Summarizing PastDraw's display of informational elements.....	118

List of Figures

Figure 1.1: Comparing versions of diagrams to track changes manually (a) and (b) and with the application of color (c).....	3
Figure 1.2: Research context. The darkly shaded oval indicates the domain of interest for this thesis.....	5
Figure 2.1: Changes are presented at two different levels: on a version-by-version basis and on the level of the individual function (Magnusson and Asklund 1996).....	12
Figure 2.2: Comparing two versions of a simple file and console I/O program with Diff.	15
Figure 2.3: In Microsoft Word (Microsoft 1983), the modification of a string of text is treated as an insertion followed immediately by a deletion of the old text.....	17
Figure 2.4: The four-column view of changes provided by Flexible Diff (Neuwirth, Chandhok, Kaufer, Erion, Morris and Miller 1992).....	18
Figure 2.5: Version differencing of Visual Age (IBM 1991).....	20
Figure 2.6: Read-wear/Edit-wear showing one category of wear. Adapted from Hill and Hollan (1992).....	23
Figure 2.7: Using multiple wear indicators to show which parts of the document have been read by different people. Adapted from Hill and Hollan (1992).....	23
Figure 2.8: Overview of a portion of different chapters of this thesis displayed with Microsoft Word (Microsoft 1983) at 50% (a) and 10% (b) of the original size.....	24
Figure 2.9: A re-creation of Seesoft (Eick, Steffen and Sumner 1992). It shows the modification requests for bug fixes made to a software project.....	25
Figure 2.10: The Theme River (Havre, Hetzler and Nowell 2000) system represents changes as rivers that flow over time.....	26
Figure 2.11: The modification of a picture illustrated by showing the modified version (left) and the original version (right with a red strike-through) in Word (Microsoft 1983).	28

Figure 2.12: The regular working view of UML diagrams (a) and the Model Integrator view (b) which is the only (indirect) support provided by Rational Rose (Rational Software Corporation 1991) to track changes.....	30
Figure 2.13: Editable graphical histories, showing the series of commands (bottom) needed to produce the final picture (top). Adapted from Kurlander and Feiner (1993)....	31
Figure 4.1: Presentation, placement and persistence of awareness display dimensions...	57
Figure 4.2: Example mock-ups illustrating several display dimensions.....	60
Figure 4.3: The Perspective wall as an example focus and context representation (Mackinlay, Robertson and Card 1991).....	63
Figure 4.4: Querying for changes from two different workspace perspectives (mockups).	65
Figure 4.5: Change history of a document in a Visual SourceSafe repository (Microsoft 1992).....	67
Figure 4.6: Display mechanism for showing changes made to the workspace outside of the application (mockup).....	68
Figure 4.7: Using position to display the number of changes (mockup).....	73
Figure 4.8: Using shape to represent changes.....	74
Figure 4.9: Using value to communicate how nodes ranked according to the number of changes made to them.....	75
Figure 4.10: Using orientation to represent change awareness information.....	76
Figure 4.11: An example where orientation to communicate change awareness information may not work (directed graphs).....	77
Figure 4.12: Employing texture to display change awareness information.....	77
Figure 4.13: Misuse of texture to represent changes.....	78
Figure 4.14: Animation of a node being deleted in a concept map editor (Gutwin 1997).	79
Figure 4.15: Using before and after comparisons to spot differences between panels (courtesy of King Features Syndicate, Inc).....	80
Figure 4.16: Different detail levels in the display of changes using text.....	81

Figure 4.17: Different approaches to filtering.....	82
Figure 4.18: An E-banking software system modeled with UML diagrams.....	83
Figure 4.19: Using basic filters to screen changes.....	84
Figure 4.20: Before (a) and after (b) versions.....	85
Figure 4.21: Filtering changes that are depicted with storyboards.....	86
Figure 4.22: Semantic filtering of change information.....	86
Figure 4.23: Hierarchy of UML package and class diagrams.....	88
Figure 5.1: Color and variations in saturation and value are used to indicate if a class has changed and by how much.....	93
Figure 5.2: Animated replay of changes (literal, situated).....	94
Figure 5.3: Overview and detailed view of changes using storyboarding.....	95
Figure 5.4: Text labels used to represent changes.....	96
Figure 5.5: Using written documentation to represent information about changes.....	97
Figure 5.6: Comparative ratings of the different display mechanisms.....	99
Figure 6.1: A basic drawing application created from the classes in the GEF (Tigris 1995).....	106
Figure 6.2: Overview of the main features of PastDraw.....	108
Figure 6.3: Text labels describing the changes.....	110
Figure 6.4: Multiple levels of detail provided with text labels.....	111
Figure 6.5: Changes to the second floor and the documentation of a specific change.	
.....	111
Figure 6.6: Text labels in the main view (a) and the document overview (b).....	120
Figure 6.7: Classifying the display mechanisms employed by PastDraw.....	124
Figure 6.8: Separately placed change mechanisms can be hard to relate to the changed object.....	126
Figure 6.9: Notifying users of changes outside of PastDraw (mockup).....	129
Figure 6.10: The new and improved PastDraw (mockup).....	131
Figure 6.11: Users can increase the granularity of panels to probe for further details (mockup).....	131

Chapter 1

Introducing change awareness

1.1 Definition of change awareness and the need for it in group work

Many projects depend upon collaboration between people. The motivating factors behind the collaboration varies from project to project. Some may require a particular combination of skills and expertise not entirely held by any one person in the project or there is simply too much work for one person to do it all. However, when multiple people make changes within a project, things can quickly spiral out of control as changes made by one person can wreak havoc on the work of others or even the entire project.

In software development projects, one strategy for dealing with this problem is *change management*. It focuses on modeling and controlling of the process of change through *configuration management*, defined as

“...the art of identifying, organizing and controlling modifications to the software being built by a programming team. The goal is to maximize productivity by minimizing mistakes” (Pressman 1997: pp. 209).

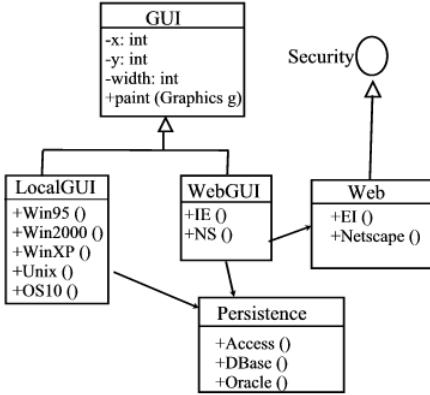
Yet, in certain situations, exacting control of the changes may not be possible. It may simply not fit in with the work culture of the group. In addition, although changes may be formally regulated, people may still have trouble keeping up with what the other collaborators are doing over time in even modestly sized projects. This is an even more challenging and time-consuming endeavor if the project is large, if it contains many interdependent parts, and if it is in a constant state of flux. In such situations, *change awareness* is crucial to project success. I define *change awareness* as:

The ability of a person to track the changes that other collaborators have made to a group project.

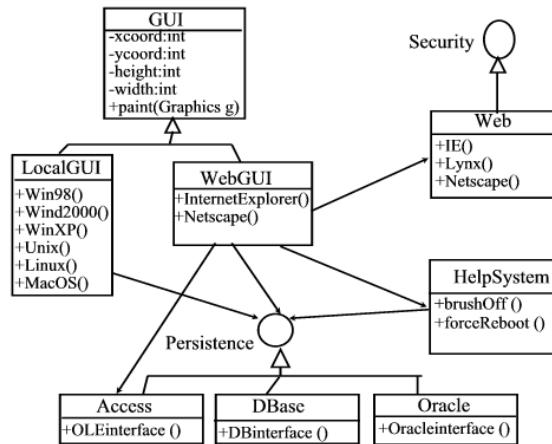
Much research has been conducted into change awareness for text-based collaborative projects (e.g., Neuwirth, Chandhok, Kaufer, Erion, Morris and Miller 1992; Hill and Hollan 1992; Eick, Steffen and Sumner 1992). Furthermore, there are many computer applications that support changes in text documents and text collections, which include version control, configuration management and file differencing systems.

In contrast, very little has been done to support change awareness in other types of documents. In particular, I believe that change awareness support is needed is in two-dimensional graphical documents and workspaces, such as those used to create figures, blueprints, concept maps and UML diagrams (Booch, Rumbaugh and Jacobson 1999; Fowler and Scott 1997; Rumbaugh, Jacobson and Booch 1999). Users of these programs often must resort to manual techniques to monitor changes. For example, Steves, Morse, Gutwin and Greenberg (2001) evaluated the usability of TeamWave (TeamWave Software Ltd. 1997), a room-based 2D groupware application. They found that one of the application's major usability problems was a lack of support for change awareness. "Participants reported that they were unable to effectively and efficiently detect modifications to artifacts made by others in the rooms they were using without the email notification scheme [a manual workaround that some TeamWave users devised to provide the information that they needed]..." (pp. 5).

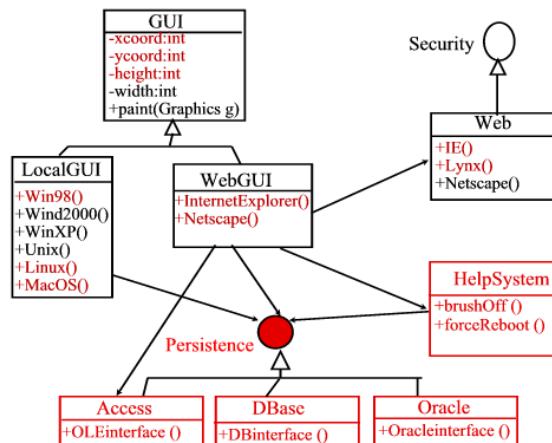
A common technique for dealing with a lack of change support is to visually compare document versions to try to spot changes and decipher their meaning. This is both a daunting and error prone task. For example, Figure 1.1 shows three versions of a UML class diagram. There are a total of 15 differences in the classes between the before and after version shown in (a) and (b).



a) A before version of a UML diagram.



b) An after version of a UML diagram.



c) An after version of a UML diagram (augmented with color).

Figure 1.1: Comparing versions of diagrams to track changes manually (a) and (b) and with the application of color (c).

Although a person may be able to eventually determine all the differences, it requires much time and effort to do so. The version depicted in (c) highlights specific blocks of changed text by the application of color. Whenever changes apply to the entire class everything is colored red. Even this simple technique better allows us to spot changes than by using manual comparisons.

1.2 Research context

Figure 1.2 illustrates different types of two-dimensional graphical applications. Two broad categories separate systems that produce bitmaps vs. those that produce structured drawings. With bitmaps users manipulate only pixels. In contrast, structured drawings have clearly distinguishable graphical objects that can each be manipulated and they have their own set of semantics. All of these graphical elements and the surface on which they sit comprise the graphical document. Structured drawings can be quite general, where people manipulate basic objects including lines, circles, rectangles, text boxes, and so on e.g., PhotoDraw (Microsoft 1995). They can also be application-specific. In concept map editors, for example, people manipulate graphs (nodes and arcs) containing ideas and semantic links (Gaines and Shaw 1995). Similarly, in UML editors people manipulate graphical elements representing classes, properties, methods and so on. Because of their semantic richness, the focus of this thesis will be these types of 2D structured drawing applications (indicated by the darkly shaded oval in Figure 1.2), and how people work together within them.

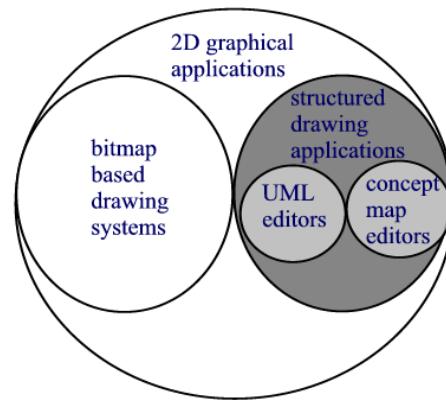


Figure 1.2: Research context. The darkly shaded oval indicates the domain of interest for this thesis.

1.3 Research objectives

My interest is in change awareness between collaborators working asynchronously at different times within 2D structured graphical editors. That is, I refine the definition in Section 1.1 to:

Change awareness is the ability of a person to track the changes that other people have made to graphical objects.

My primary goal in this thesis is to describe this concept of change awareness in detail, and what support must be provided by software for dealing with it. This main goal comprises four sub-goals.

- 1) To explore ideas in change awareness and to gain a hands on understanding of it. I will do this by extending an existing graphical 2D structured drawing program to track and display change awareness information.
- 2) Produce a conceptual framework for understanding change awareness in 2D graphical collaborative workspaces, where multiple collaborators interact over the space at different times. I will do this by expanding Gutwin's framework on workspace awareness (Gutwin 1997), which focused on investigating real time interactions, to include in detail how people can stay aware of changes over time.

3) Develop a set of basic concepts to guide the display of change awareness information.

This includes the visual representation of change information as well as mechanisms for filtering this information. I will do this by applying previous information visualization research e.g., Card, MacKinlay and Shneiderman (1999) as well describing ideas that are unique to change awareness.

4) Demonstrate the value of the framework and display concepts. I will do this by critiquing the system (implemented in the first goal) according to the criteria laid out in the framework (developed in the second goal) and the display concepts (developed in the third goal).

1.4 Outline of this thesis

The remainder of this thesis is laid out as follows. In Chapter 2, I describe previous research relevant to handling the problems of change in collaborative environments. I begin by discussing some well-established methods for handling change such as version control systems. I continue by describing how change is handled in sequential text documents. Finally, I conclude Chapter 2 by describing the almost non-existent support for change awareness provided in graphical environments.

In Chapter 3, I lay out a conceptual framework that can be applied to handling change awareness for 2D graphical and collaborative work environments. This includes the basic categories of questions that people may ask about changes made in the workspace, and the informational elements that are relevant to answering those questions.

I then describe in Chapter 4 the different concepts involved in the representation of change awareness information so that it can be done in an effective manner. This requires consideration of: the dimensions for displaying change information; ways of making the information readily available and easily comprehensible; mechanisms for displaying changes; and techniques for filtering information about changes.

During Chapter 5, I briefly digress in order to describe the results of a formative pilot study that was conducted to investigate some different ways of representing change information. The results of this study are not meant as a definitive guide as to how to represent this type of information. Rather it highlights some potential mechanisms that deserve further investigation.

In Chapter 6 I use the framework laid out in Chapter 3 and the display concepts discussed in Chapter 4 to critique an example 2D structured drawing program that I augmented with change awareness. This system was developed before the framework, and its evaluation provides useful insights about the value of the framework and display concepts as high-level design guides and as tools for assessing change awareness support systems.

I close this thesis in Chapter 7 by describing how I met the goals listed in this introductory chapter and list my research contributions. I also include some suggestions for areas of future research into change awareness.

Chapter 2

Previous research into change awareness

In this chapter, I describe existing approaches and system designs used to manage or display changes in collaborative projects. I start with early mechanisms for controlling and tracking changes between versions of text files and between collections of revision controlled files. Next, I survey popular techniques for displaying changes, as used by text file differencing systems built into both commercial and research word processing systems. These approaches work mostly with text. In the final section, I briefly present the few systems that support change in two-dimensional graphical systems.

2.1 Version control and configuration management systems

In this section, I describe a progression of standard systems that track change at a document or file level. Most work best with ASCII text documents (ASCII-based source for programs) and collections. I cover basic concepts including: manual and automatic maintenance of changes; conflict management; interdependencies between files and modules in a collaborative project; version and change granularity; providing a high level overview of all changes; and the management of binary files.

2.1.1 Manually tracking changes through backups

When people have no automated change support, they usually make backups of files as a manual way to save versions and track changes. Backups are useful both in individual and group projects: each time that a file is changed, another copy of the file would be made. This is done automatically for users of the VAX/VMS operating system (Compaq 1975). In both cases (manual and automatic), when someone wants to examine the changes made over time, they simply restore different backup files and visually compare older versions with current versions (Tichy 1991).

Not surprisingly, this technique has problems. In projects involving many changes by many people, backup versions accumulate so quickly that people have trouble managing the collection. They end up spending much of their time browsing and deleting useless files (Tichy 1991). As well, they often find it difficult to understand exactly what has changed between the many backup sets for there is no automatic way of distinguishing between versions containing trivial *vs.* important changes or of even finding the changes. Additionally, because manual backups requires that attention be diverted from the main task of making changes, people often neglect or forget to make them, even when their changes may have significant and widespread ramifications.

2.1.2 Version control systems

Version control systems automate and enhance the process of manually tracking progressive versions of documents and their changes. Version control is defined as the task of keeping the versions and configurations of a software system well organized (Tichy 1991). That is, these systems help people save and track different versions of individual documents (Magnusson and Asklund 1996). Example systems include the RCS Revision Control System (Tichy 1991), IBM's Clear/Caster system (Brown 1970), AT&T's SCCS Source Code Control System (Rochkind 1975), CMU's SDC Software Development Control System (Habermann 1979), and Digital Equipment Corporation's CMS Code Management System (DEC 1982).

All version control systems work by separating the editing of a document from the version control of a document. The first revision (version) is created from the document, which effectively “freezes” it. The frozen version can no longer be changed, and so editing it implicitly creates a new revision (Tichy 1991).

Version control systems overcome some of the problems found with the manual comparison of document versions by displaying differences through an algorithm such as *Diff* (Hunt and McIlroy 1975). *Diff* produces “...a small, line-based delta [a sequence of commands needed to transform one file to another] between pairs of text files” (Tichy

1991: pp. 7). Differences are determined on a line-by-line basis so that if only a single character in a line is changed, the program would flag the entire line as being changed. This approach will be discussed further in Section 2.2.

Version control systems also try to minimize conflicts that can occur when two or more people try to edit the same version. That is, the system controls the process of change to, “...detect conflicts and prevent overlapping changes” (Tichy 1991: pp. 1). This control is usually implemented through a “locking” mechanism that only allows one person to edit a version at a time (although this mechanism can be bypassed if so desired).

While useful for managing simple file sets, version control systems are limited. In an active project of moderate size that consists of interdependent parts, managing the changes of many authors remains daunting. Changes made in one part of the project can result in further, unexpected, changes in another part. It is hard to predict all of the possible consequences of a change made to complex systems without computer assistance (Luqi 1990; Arango, Schoen, Pettengill and Hoskins 1993). As a result *configuration management* systems were developed, which will be discussed in the next section.

2.1.3 Configuration management systems

Version control systems work best for simple file collections with few interdependencies and only a modest number of authors. Projects involving multiple authors and many interdependencies are better served by sophisticated configuration management systems, such as CVS (Berliner 1990), Visual SourceSafe (Microsoft 1992) or the IDE (integrated development environment) VisualAge (IBM 1991). As summarized by the British Standards Institute (1984), configuration management identifies the total configuration (i.e., the hardware, firmware, software, services and supplies) at any time in a system’s life cycle, together with any changes or enhancements that are proposed or are in the

course of being implemented. Thus, it allows changes to be traced through the lifecycle of a system or across a group of associated systems.

The main purpose of configuration management systems is to keep developers aware of the status of different software components at any point in time. Without awareness of all the components in a configuration, each modification made could result in disaster because the consequences of the change are unknown (Allan 1997).

2.1.4 Limitations of version control and configuration management systems

The first drawback of some version control and configuration management systems is that the number of changes between versions is quite large. Most version control and configuration management systems tend to be employed in cases where the number of changes across files is numerous and where the granularity of changes between files versions is quite coarse. As described by Sobell (1999) “...when you are fixing a collection of bugs in a file, you should fix each one and completely test it before recording the changes in a delta (version). This technique saves you from having deltas that reflect incomplete, transitional changes in the history of a file” (pp. 581).

When files contain well-defined parts, it may be more helpful to track changes of these parts as well as of the whole file. This is what is done in *fine-grained version control*, a term that was introduced in Orwell (Thomas and Johnson 1988). It refers to version control at the function or method level rather than at the class or file level (Magnusson and Asklund 1996). This means that programmers are able to track changes between different versions of methods, as well as at the file level.

A version control and configuration management system that does allow for such a precise degree of version control is COOP/ Orm (Magnusson and Asklund 1996) shown in Figure 2.1. The upper portion of the figure labeled “versions” shows the version history of a software project. We see seven different versions of the system. The main branch consists of version 1, 2, 4, 5, and 7 while versions 3 and 6 are side branches. The

arrows going from version 3 to 5 and 6 to 7 indicates a merging of different versions. In the bottom of the figure, versions 2 and 7 are being compared.

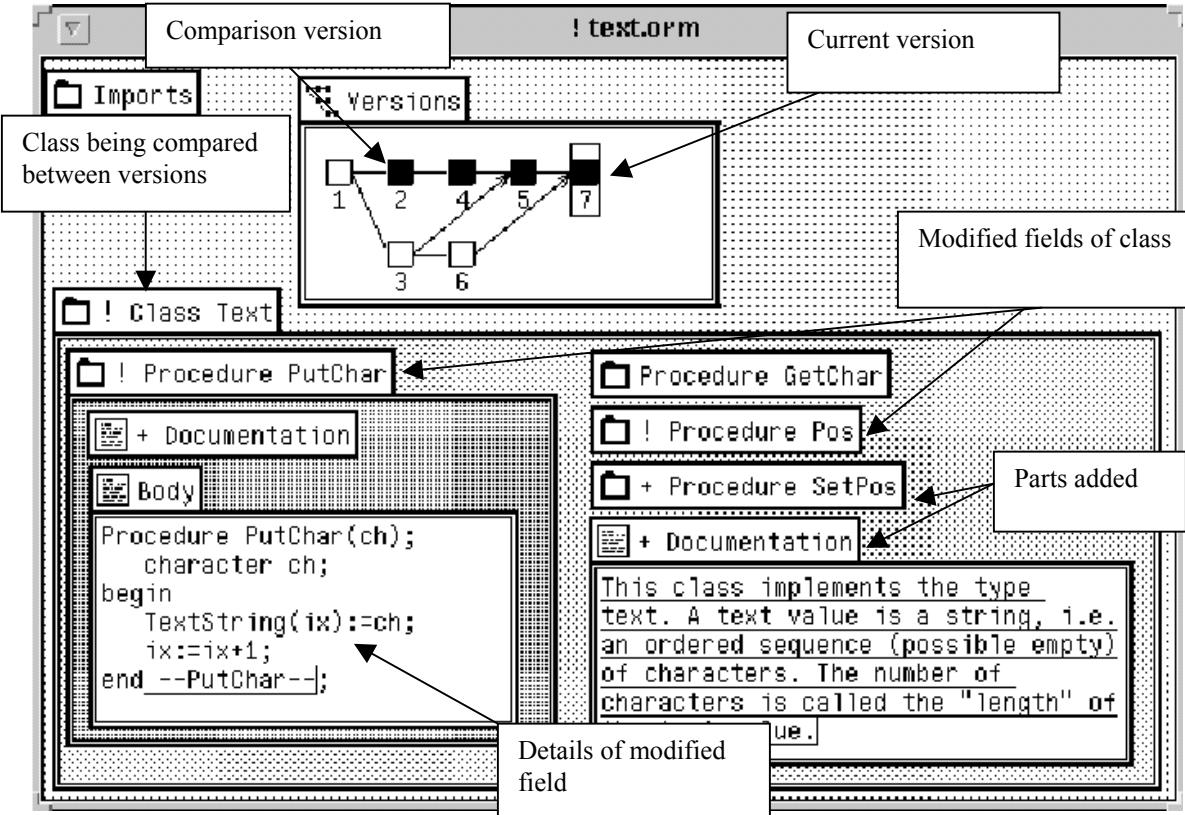


Figure 2.1: Changes are presented at two different levels: on a version-by-version basis and on the level of the individual function (Magnusson and Asklund 1996).

We see a folder in the lower part of the figure showing how a part of the system, the class “Text” differs, between version 2 and version 7. There is a “!” beside the name of the class indicating that this class was modified somehow between version 2 and version 7. The different procedures for “Text” and the documentation for the class are shown within the folder for the class. We also see that documentation has been added to the class between versions, and that the procedure “SetPos” was added; both additions are indicated by the “+”, while the modified procedures “Pos” and “PutChar” are indicated by the “!”. Procedure “GetChar” is identical between the versions so that there is no symbol beside the name of the method.

Within the folder for class Text, there is another folder that provides the details of how procedure “PutChar” has changed i.e., documentation has been added to the procedure. We see that the actual code for PutChar is displayed in the folder called “Body” where additions are depicted with underlining and deletions by strikethroughs.

A second drawback of many version control and configuration management systems is that they depict only change details at a file-by-file level. With many changes, it might be useful for a person to get an overview of them to help one decide what documents and document parts that one should examine in detail. Yet there is no automated support for tracking changes to a document as a whole let alone for a collection of related documents. Doing so could be a very useful mechanism for helping a person track changes in (say) a software system that consists of many constantly changing source text files.

A third problem is that most version control systems list changes out of context, usually through a report presented separately from the changed documents. This means that one must look at a report item and try to figure how it relates to particular changes in the document itself e.g., as in the case of Diff (see Section 2.2). Context is often missing, and people must ‘navigate’ between the report and document. COOP/Orm is an exception. As Figure 2.1 shows (in the folder called “Body”), the system does display changes inside the document (source code) right at the point where the change was made. This way, the reader can get a better sense of the context supporting the change i.e., the operations performed by that portion of the code, and other changes made near it. Within this context different types of changes are indicated with different mechanisms such as underlining and strike-throughs.

A final serious problem is that many of the systems mentioned so far are fairly old and were developed for managing ASCII-based text files. Modern computers, however, deal extensively with non-ASCII, binary files containing *rich text* (text with formatting instructions), music, photographs, and images. Although some systems such as CVS (Berliner 1990) and Visual SourceSafe (Microsoft 1992) do allow for rudimentary

management of binary file versions, they work best for text files. This is because these systems cannot compute or display the actual differences between two binary files. Most use a text-based differencing algorithm that cannot make sense of binary data whose meaning depends largely upon the application and even the hardware that is used to create it. Thus they can only report that the versions differ, which provides little useful information. To display differences between binary files in a meaningful way, the version control system would need to know the structure of the binary files it handles. Given the prolific use of binary files, often with proprietary or undocumented internal structures, it is simply impossible for a version control system to handle all binary files in a robust and generic way. Consequently, people now rely on explicit documentation of changes made by the author to fully understand the differences between versions of a binary file, or they must hope a particular application knows enough about the binary file to meaningfully present information about changes.

2.2 Displaying change in text-based systems

In the first subsection below I describe the early systems for tracking and managing changes such as Diff. As mentioned in the previous section, many of these systems would represent changes separately from the changed documents, which often made change tracking difficult. In the next subsection I describe some of the later systems, such as Word (Microsoft 1983), which would imbed information about changes right in the document itself. The next subsection deals with cases when a document becomes so large that a ‘bird’s-eye’ overview of changes becomes necessary.

2.2.1 Text based differencing

Most systems organize change information into versions and track where (e.g., the line and character position) changes have occurred. A typical example system is Diff (Hunt and McIlroy 1975), which can be used to compare and display line-by-line changes between two different versions of ASCII files. If these systems are to be useful, a person must also be able to visually compare differing versions so that they can see and

understand exactly what changes were made. As will be seen in the example illustrated in Figure 2.2, this is one of the major drawbacks of Diff as a tool for supporting change awareness.

```

2,4c2,4
< This is a small C program. It prompts the user for the name
< of a file and then it reads from the file, a character at a time
< and displays the resulting values to the standard input.
---
> This is the C++ port of the C program file.c. It uses
> the stream classes found in fstream to perform all of the file
> input and output.
12,15c12,13
< /* Preprocessor directives */          */
< #include <stdio.h>
< #define FILESIZE 80
< #include <assert.h>
---
> // Preprocessor directives
> #include <fstream.h>
17a16,18
> // Program constants
> int const FILESIZE = 80;
>
21,24c22,25
< /* Variables that are local to main */      */
< FILE *f = NULL;
< char tempChar = -1;
< char fileName[FILESIZE];
---
> // Variables that are local to main
> ifstream fin;
> char inputFile[FILESIZE];
> char tempChar;
26,36c27,33
< /* Prompt user for inputs */                */
< printf("Enter in the name of the input file:");
< gets(fileName);

```

Figure 2.2: Comparing two versions of a simple file and console I/O program with Diff.

The output of Diff, seen in the figure, deserves some explanation. Changes between files are shown on a line-by-line basis in the form of a linear list. Diff outputs a left angled bracket “<” in front of lines from the first version that do not match any lines in the second version. The right-angled bracket “>” is in front of lines in the second version that do not match any lines in the first version. Lines common to both are not displayed. The problem is obvious: changes are shown out of context. Although Diff produces compact output, readers can find it hard to understand the changes because they must recall from memory what the surrounding lines of text are like in order to recover the context of the changes. Often readers must relate Diff output to the two different source files.

Furthermore, if only one character in a line differs between versions, Diff will show the entire line as being different between versions leaving the reader to manually compare the lines in order to determine exactly how they differ. Not surprisingly, if many changes have been made between versions, the reader ends up spending much time reading and re-reading Diff output to determine what has changed. In addition, Diff does not output any additional cues to help differentiate between additions, deletions, modifications or authorship, or the sequence of changes. Because of all of these limitations, one finds Diff unsatisfactory for even the simple example given in Figure 2.2 that has few changes to be compared.

With a large system that is being modified constantly by many different programmers, tracking changes on a line-by-line basis is a daunting task. Not all the changes will be relevant to a given programmer so that each person must wade through them all in order to determine which ones apply to his or her work. Eventually the issues associated with displaying change information in text-based collaborative environments began to receive research attention, and as a result later systems improved this display. These will be discussed shortly in Subsections 2.2.2 and 2.2.3.

2.2.2 Extensions to the differencing algorithms: presenting changes at the document level

A better approach to differencing is to display rich change information within the text document itself. Some time after the advent of graphical word processors that use rich text, non-programmers demanded effective ways of tracking and viewing changes between document drafts. Consequently common word processors such as Microsoft Word began to include change awareness techniques using in-line change displays, change indicators and annotations. The types of changes tracked are the addition, deletion or modification of items in the document. Figure 2.3 shows a Word document with the change tracking function turned on. We see that modifications are treated as an insertion immediately followed by a deletion. We also see a vertical bar in the left margin beside lines that have changed, which helps people spot lines with small changes.

Text that is the same between versions is rendered in the original black; text that differs between versions is rendered in different colors, one for each editor. The example in the figure highlights changes by coloring the changed text red. Text added to the version is rendered with an underlined typeface, while text that has been deleted is rendered with a strike-through typeface. Changed text that has been annotated with comments by an author is highlighted in yellow, and mousing-over it will reveal the annotation in a small, transient, pop-up window.

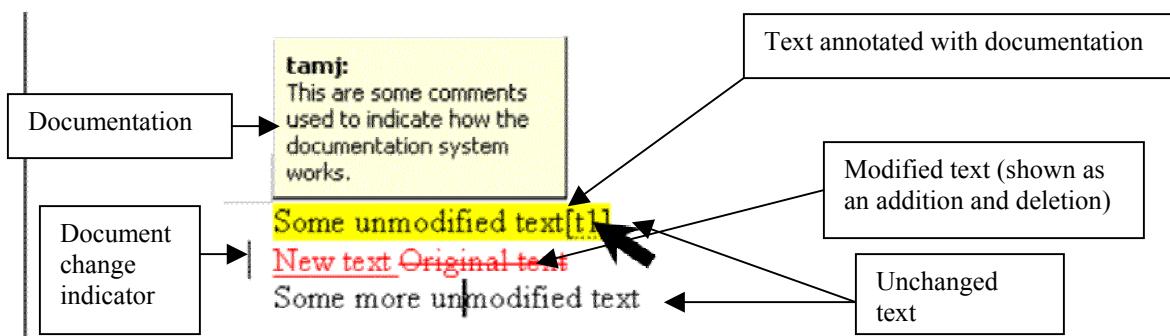


Figure 2.3: In Microsoft Word (Microsoft 1983), the modification of a string of text is treated as an insertion followed immediately by a deletion of the old text.

Unlike version control systems, which compare saved snapshots, Word allows the reader to track changes in real time. As a change is made to the document, a corresponding change indicator will be immediately displayed to the user. In the older version control systems, one had to leave the text editor and run a separate program to track changes.

A more sophisticated system that displays changes inline is Flexible Diff (Neuwirth, Chandhok, Kaufer, Erion, Morris and Miller 1992), illustrated in Figure 2.4. Flexible Diff is an extension of the PREP editor (Cavalier, Chandhok, Morris, Kaufer and Neuwirth 1991; Neuwirth and Kaufer 1989; Neuwirth, Kaufer, Chandhok and Morris 1990). In this figure, we see how four columns are used to communicate the nature of a change. The original and modified text are in the first and second columns respectively. Column 3 shows only the differences between these versions while Column 4 shows annotations added by the author that explain the changes.

This last column of annotations was included to alleviate the frustrations that authors would sometimes experience when they encountered unexpected changes made by other co-authors (Cross 1990). Annotations also help draw a reader's attention to a particular change (Neuwirth, Chandhok, Kaufer, Erion, Morris and Miller 1992). If we go back and compare the output from the original form of Diff in Figure 2.2 with the output of Flexible Diff in Figure 2.4, it is obvious that this four-column format makes it much easier for the reader to find, track and understand changes. The cost is that the original document format (e.g., its formatting, use of white space, etc.) is lost in this view.

Original	Revision	Comparison	Explanation
Way to decrease coordination difficulty is to communicate less distracting information and more relevant information.	One way to decrease coordination difficulty is to communicate less distracting information and more relevant information - in effect, lowering the "signal-to-noise" ratio.	Way One way ... - <i>in effect, lowering the "signal-to-noise" ratio</i>	"signal-to-noise" - is this an understood phrase?
The approach is incorporated in the "work in preparation" (PREP) Editor, a word processor being developed to study and to support collaborative writing processes.	The approach to comparison is incorporated in the "work in preparation" (PREP) Editor, a writing environment being developed to study and to support collaborative writing processes.	to comparison ... <u>a writing environment</u> <u>a word processor</u> ...	I don't want to position PREP as a word processor.

Figure 2.4: The four-column view of changes provided by Flexible Diff (Neuwirth, Chandhok, Kaufer, Erion, Morris and Miller 1992).

The “flexible” part of Flexible Diff refers to the fact that this algorithm, unlike its predecessors, can vary the granularity in the display of changes. A reader can specify the “grain size” (Neuwirth, Chandhok, Kaufer, Erion, Morris and Miller 1992) as the length of a text string that must be changed before the change is reported. Within this grain size, the reader can also specify that changes should be shown only where some proportion of the grain was changed. For instance, if the reader chooses to display changes at the 100% level of the word grain size, then he or she will see change information displayed

for words that are not exactly the same between versions. Setting the percentage of grain size changed below 100% means that some changes may not be displayed to the reader. The reader can also choose to ignore changes made to white space (spaces, tabs, new lines). Combined, these features give the reader an opportunity to filter out the display of trivial changes about which he or she need not be concerned.

This filtering of changes is important. Nachbar (1988) argues that it is not always appropriate to report all changes for some types of tasks. Most models of reading assume that reading requires that the reader must expend some of his or her attention in order to derive meaning from text. Since a reader's attention is limited, it must be allocated among tasks (Samuels and Kamil 1984). When multiple tasks demand more attention than one can afford, the tasks cannot be performed in parallel and instead one must allocate his or her attention carefully with some form of "attention switching" (Neuwirth, Chandhok, Kaufer, Erion, Morris and Miller 1992). However, this attention switching puts a heavy load on short-term memory and may interfere with recall (Kahneman 1973).

Simply displaying change information invites the reader to pay attention to it: this could *distract* the reader, and steal this person's attention away from his or her primary task e.g., when he or she is more concerned with reading the document rather than tracking changes between different versions. Lesser and Erman (1980) define distraction in this context as the degree to which a person's focus can be shifted. Neuwirth, Chandhok, Kaufer, Erion, Morris and Miller (1992) further specify that "positive distraction shifts the agent to tasks that are more useful; negative distraction shifts the agent to tasks that are redundant or diversionary" (pp. 149). Thus, they advise that the only changes that should be reported are ones that "...will reduce negative distraction and increase positive distraction" (pp. 149). Unfortunately, whether the display of a given change will increase positive distraction depends largely on the reader's goals for reading the document. It is for precisely this reason that Flexible Diff that permits the reader to adjust the degree to which information about changes are displayed.

VisualAge (IBM 1991) is an IDE (Integrated Development Environment) produced by IBM for writing software in many different programming languages such as Java (Sun Microsystems Inc. 1996), C++, Smalltalk etc. Its integrated version control and differencing function also overcomes some of the shortcomings of Diff. In the top window of Figure 2.5, two different versions of class are compared, and unlike Diff, an indicator is provided next to classes, methods and data to indicate the type of change that occurred between versions: removals, additions and modifications are tagged by ‘Removed’, ‘Added’ and ‘Source changed’ respectively.

VisualAge also allows changes to be viewed and filtered at various levels of the hierarchy: at the package, class or method level. In the case of viewing changes, multiple lower-level changes are combined and described as a modification. For example, if a person were viewing changes of a package that had some classes added and deleted, VisualAge would describe the change made to the package as a modification. Users can then select particular items for detailed viewing.

Element	Differences
C ChangeIcon	Removed
C ChangeStorage	
M addChange(Change)	Source changed
M displayAnimation()	Source changed
M lastElement()	Added
M restartAnimation(int)	Added
M size()	Added
C CmdSelectAnimations	

```

** 
*/
public void displayAnimation()
{
    //this.animator.animateChanges()
    animator = new DisplayAnimation(
        this.animator.start());
}

```

```

** 
*/
public void displayAnimation()
{
    this.animator.animateChanges()
}

```

Figure 2.5: Version differencing of Visual Age (IBM 1991).

For modifications, Visual Age highlights the bottom window for each individual change, one at a time. In the top of Figure 2.5, the user has selected the method ‘displayAnimation’, which has had its source changed. The bottom left part of the figure shows the source code for the newer version of this method while the bottom right part of the figure shows the source code for the older version. This change information is presented in an in-line fashion as opposed to the separate fashion used by Diff. For example, the figure shows that there are three differences in the method “displayAnimation”, where some code was ‘commented out’ (by preceding it with a pair of backslashes), a variable was initialized, and a function call was added. The figure shows how the first part of a line is highlighted (the slashes) indicating that this is the part of the line that is somehow different between versions. Notice, however that Visual Age does not explicitly indicate that this line of documentation was added between versions (in the bottom part of window). Viewers must compare the two versions themselves in order to make this determination.

In summary, all these systems show how differences are better displayed. We saw how some (but not all) systems show changes within the text, allow side-by-side comparison of changed items by marking changes, include author annotations, and allow a reader to selectively adjust the level of how changes are filtered and displayed.

2.2.3 Presenting an overview of changes

Many of the systems discussed so far have focused on displaying the individual changes within a document. The specific changed parts of a document (paragraphs, sentences, words or characters) are marked with some form of additional information to communicate the change. What is missing is a way to communicate an overview about changes made to the entire document, which is necessary if one wants to get a general sense of what has changed and where one should start looking for details in a large document or in a collection of related documents.

One approach shows readers a graphical overview of what has changed. A good example is Hill and Hollan's (1992) 'read-wear' and 'edit-wear' overview that uses the metaphor of how objects in the real world tend to wear out from use. Figure 2.6 illustrates their "attribute-mapped scroll bars" (Wroblewski, Hill and McCandless 1990), where indications of wear are displayed as marks within a scroll bar. The marks are mapped onto a document in a position relative to the line that was changed. The more that a region of a document has been read or edited, the longer the mark will be. Since the length of the scroll bar matches the length of the document and relative positions in the scroll bar correspond to a relative position in the document, the scroll bar itself provides a rough birds eye overview of the quantity of changes and where changes within a document are located. As well, multiple categories of wear indicators can be shown in different columns of the scroll bar to communicate different information, as illustrated in Figure 2.7. Here, we see 3 columns, each displaying the read history of an individual author.

Although attribute-mapped scroll bars are useful for showing a compact overview of where changes in a document lie, they do not communicate how the document changed (e.g., what words differ) or why (i.e., they do not give authors and editors the chance to annotate the document with the reasons behind changes). One would thus expect this to be integrated within another system that does provide this information.

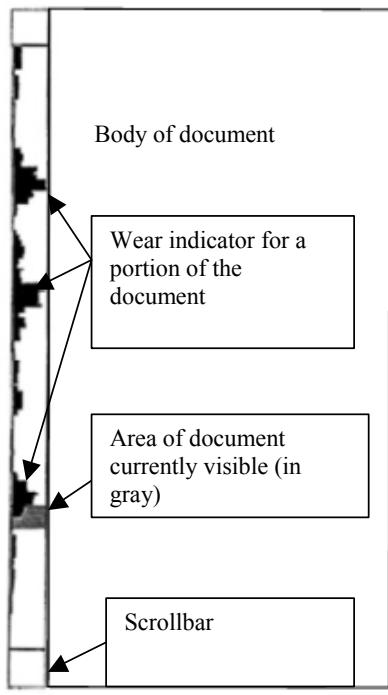


Figure 2.6: Read-wear/Edit-wear showing one category of wear. Adapted from Hill and Hollan (1992).

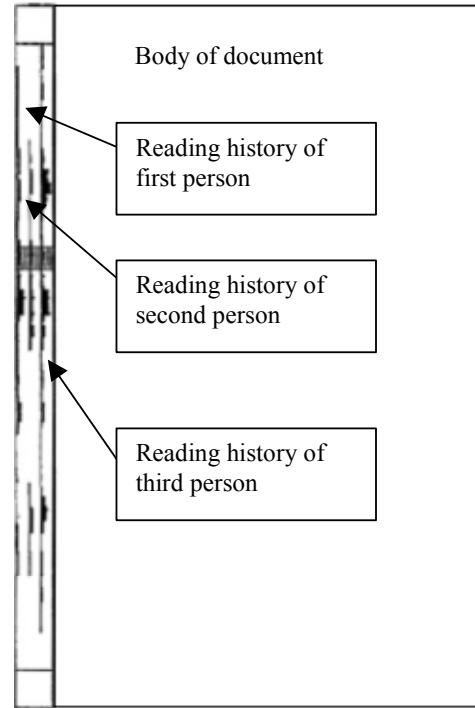


Figure 2.7: Using multiple wear indicators to show which parts of the document have been read by different people. Adapted from Hill and Hollan (1992).

A different approach to gaining an overview uses zooming. We previously saw how Microsoft Word provided a detailed in-line display of changes. Word also provides the reader with the ability to see an overview of a long document simply by letting a person ‘zoom-out’. While no additional change information is supplied, the differing colors used for fonts and graphics provide a sense of what has altered. For example, in Figure 2.8a, we see two pages of Chapter 1 of this thesis, where pages are shown at 50% of its original size. Figure 2.8b shows another chapter at 10% of the original document size. Although many more pages can be represented with a smaller document size the color indicators used to represent changes are smaller and thus are more likely to be missed.

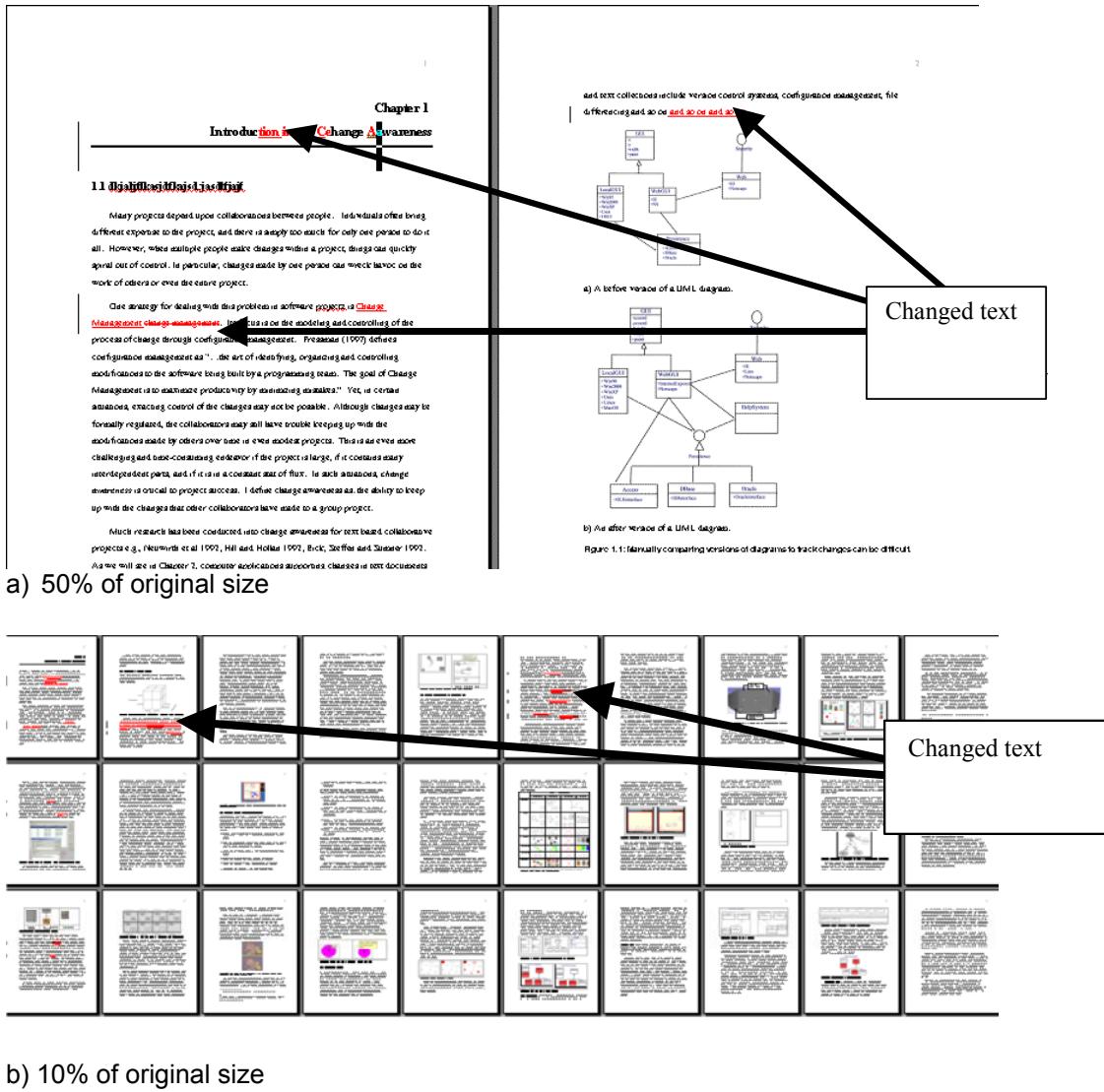


Figure 2.8: Overview of a portion of different chapters of this thesis displayed with Microsoft Word (Microsoft 1983) at 50% (a) and 10% (b) of the original size.

The Seesoft system better leverages a zoomed out overview of changes made over a collection of related documents (Eick, Steffen and Sumner 1992). Illustrated in Figure 2.9. Seesoft provides an overall view of a software system in a “reduced representation” of the code. Each column of the overview represents a single file; each line in a column represents a line in the file. The length of each line in the column shows the indentation and line length. The height of each column shows the size of the file and files that are too large to represent in a single column are wrapped to multiple columns. This

overview representation of the project is similar to the attribute mapped scroll bars (Wroblewski, Hill and Mccandless1990) except we can now see several documents at a time (up to a maximum of 50,000 lines of code or so).

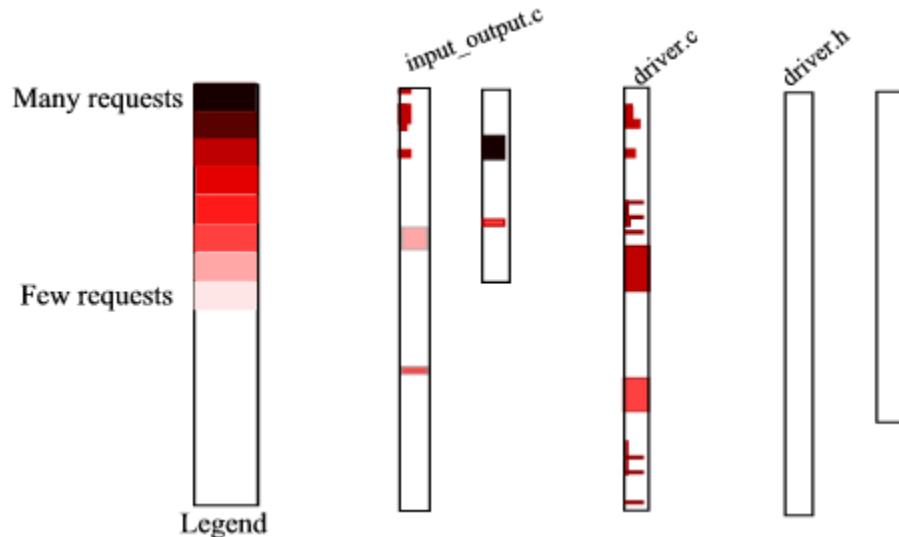


Figure 2.9: A re-creation of Seesoft (Eick, Steffen and Sumner 1992). It shows the modification requests for bug fixes made to a software project.

Unlike attribute-mapped scrollbars, users can see a detailed view of particular lines of code on demand. Whenever a user wanted to read the code that corresponded to the parts of the overview display, he or she could do so by clicking on a column. This would pull up a magnification window, which would show the actual code.

Seesoft adds value to the bird's eye view by using line coloring to indicate characteristics of the code represented, such as its age, the developer responsible for it, or the number of times it has been tested. For instance, in Figure 2.9, the brightness of the color red (as chosen through the legend on the left) is mapped to the number of modification requests made to the code. Thus we see those portions of the project shaded dark have received more modification requests than those that are more lightly shaded.

Using color, Seesoft only communicates one characteristic of the code at a time. However, the system does allow the user to change quickly the mapping between color and the different types of statistics through *Direct Manipulation* techniques

(Shneiderman 1983) i.e., the legend on the left can be manipulated to represent other attributes.

The systems seen so far portray fairly literal representations of change. The Theme River prototype (Havre, Hetzler and Nowell 2000) differs as it abstracts the minute details of changes. This system aggregates all changes made to a collection of documents and displays them as trends that are taking place in the collection over time (Figure 2.10). Theme River is based upon the principles of the Gestalt school of Psychology and the belief that “the whole is greater than the sum of the parts” (Koffka 1935). It capitalizes on the notion that people can use their perceptual skills to see high-level relationships between multiple related documents. Its visualization uses a metaphor of a river that travels horizontally across the screen (Figure 2.10). Time varies along the x-axis, while the strength of a trend (e.g., the number of times that a particular word appeared in a collection of documents) is shown along the y-axis. This visualization provides quick and aggregated representations of the rise and fall of trends (changes over time).

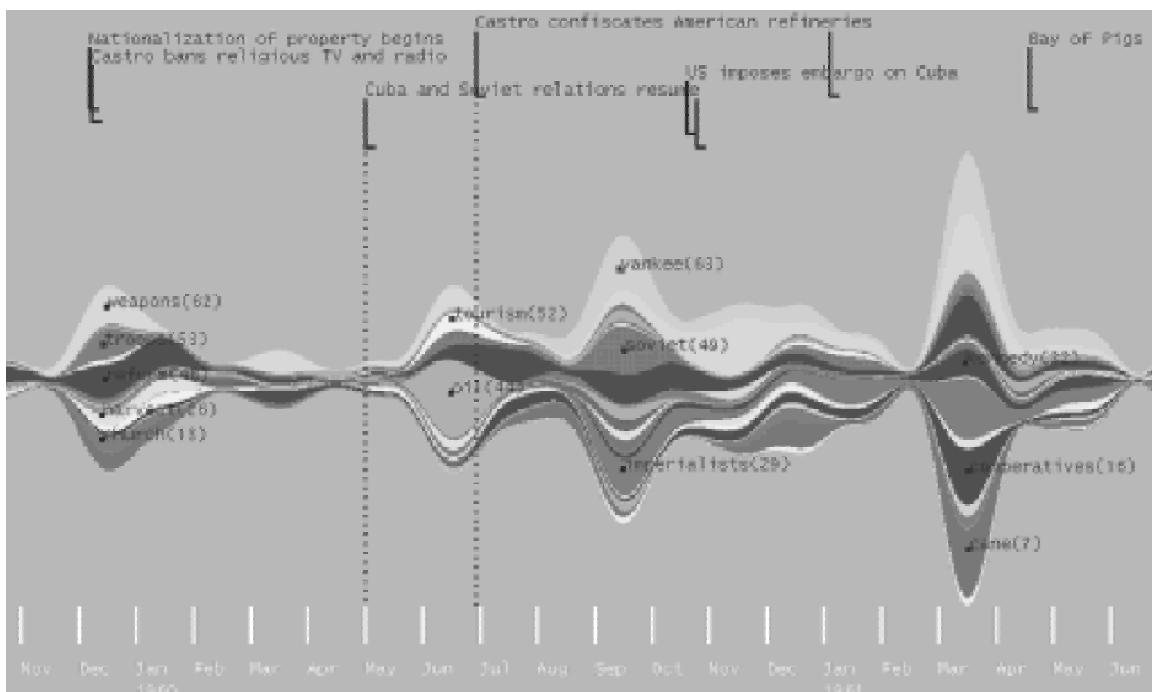


Figure 2.10: The Theme River (Havre, Hetzler and Nowell 2000) system represents changes as rivers that flow over time.

Usability tests found that while users liked abstracting from individual documents to the collection of documents as a whole, they also wanted to see individual documents on demand (Havre, Hetzler and Nowell 2000). Consequently while Theme River and perhaps other overview visualizations are useful for communicating high-level change information, other levels of detail are required. This follows Shneiderman (1996), who suggests in his InfoVis mantra, overview first, details on demand.

2.2.4 Existing portrayals of changes in text-based systems

Three key points must be kept in mind with regard to the systems I have discussed thus far. First, the representation mechanism used to display and thus communicate rich change information is an important part of the system and has a tremendous impact on the usability of the system. Second, because modern text editing systems no longer work exclusively with text, a change tracking system must also deal with non-text elements (e.g., formatting, pictures) in documents. Third, changes should be displayed on multiple levels: from the *macroscopic* perspective of the entire collection of documents or individual document right down to a *microscopic* view of changes shown on an item-by-item basis. Ideally, the granularity of changes shown should be easily and rapidly adjustable in order to suit different change-related tasks.

2.3 Change awareness in existing two dimensional graphical systems

All of the systems discussed so far deal primarily with sequential text. There has been appallingly little research into the problems associated with tracking changes made to two-dimensional drawings. This is even more astounding in light of the fact that graphics are used in a wide variety of applications. For example, graphical editing packages such as paint and structured drawing packages are used pervasively for a wide variety of familiar tasks such as constructing technical drawings, organizational charts, network diagrams, flow charts or architectural diagrams (Kurlander 1993). Less familiar are concept maps, a semantically rich method of visual communication that has "...a history of use in many disciplines as a formal or semi-formal diagrammatic technique"

(Gaines and Shaw 1995: pp. 323). Similarly, UML diagrams are the standard design and modeling language for software (Booch, Rumbaugh and Jacobson 1999; Fowler and Scott 1997; Rumbaugh, Jacobson and Booch 1999). In groupware systems such as TeamWave (TeamWave Software Ltd. 1997) a room-based metaphor is used to spatially relate and reveal the rich, natural social interactions of the everyday world within collaborative virtual work (Greenberg and Roseman 2001).

One example system that provides only crude change awareness support for graphics is Microsoft Word. Despite the rich support for text, Word has poor change support for images. Usually, the entire image is treated as though it were a single character, and when any change—large or small—is made to the picture, the entire picture is rendered with a change indicator similar to those applied to text (Figure 2.11). As with the case of changes to text, there is a vertical line added to the changed image to better indicate which part of the document was changed. Because the particular example shown is a vector graphic, Word also shows a small gray “ghosted” shadow to indicate where the two images differ: (the right peg leg of the cartoon character). With other types of images, the reader is left to his or her own devices to determine how the graphic has changed.

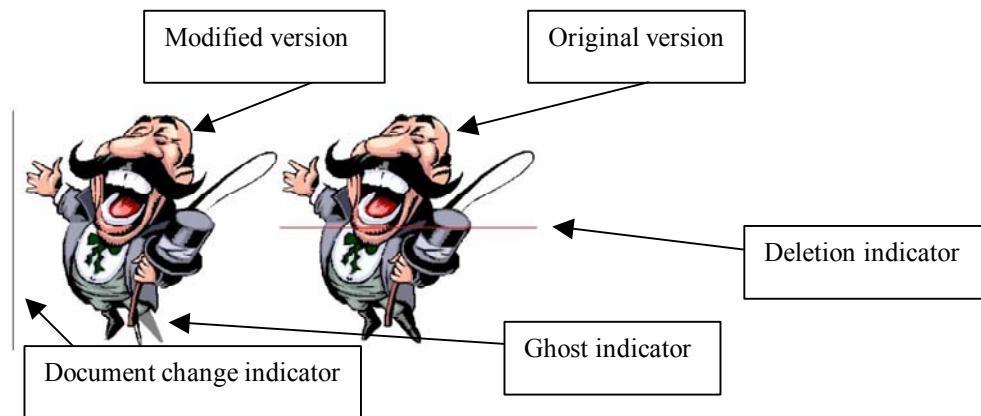
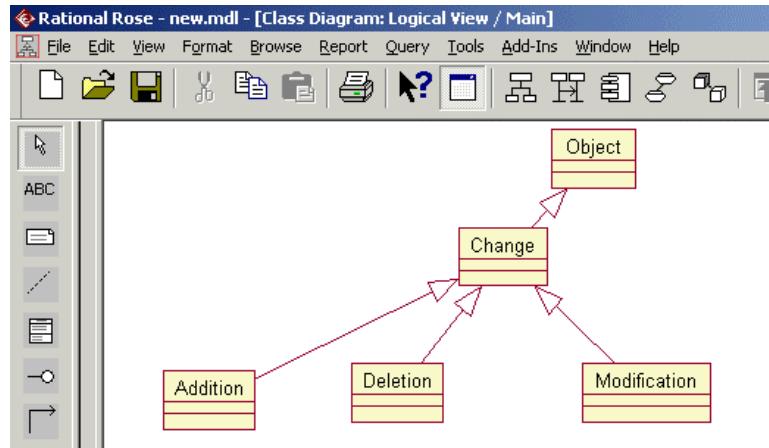


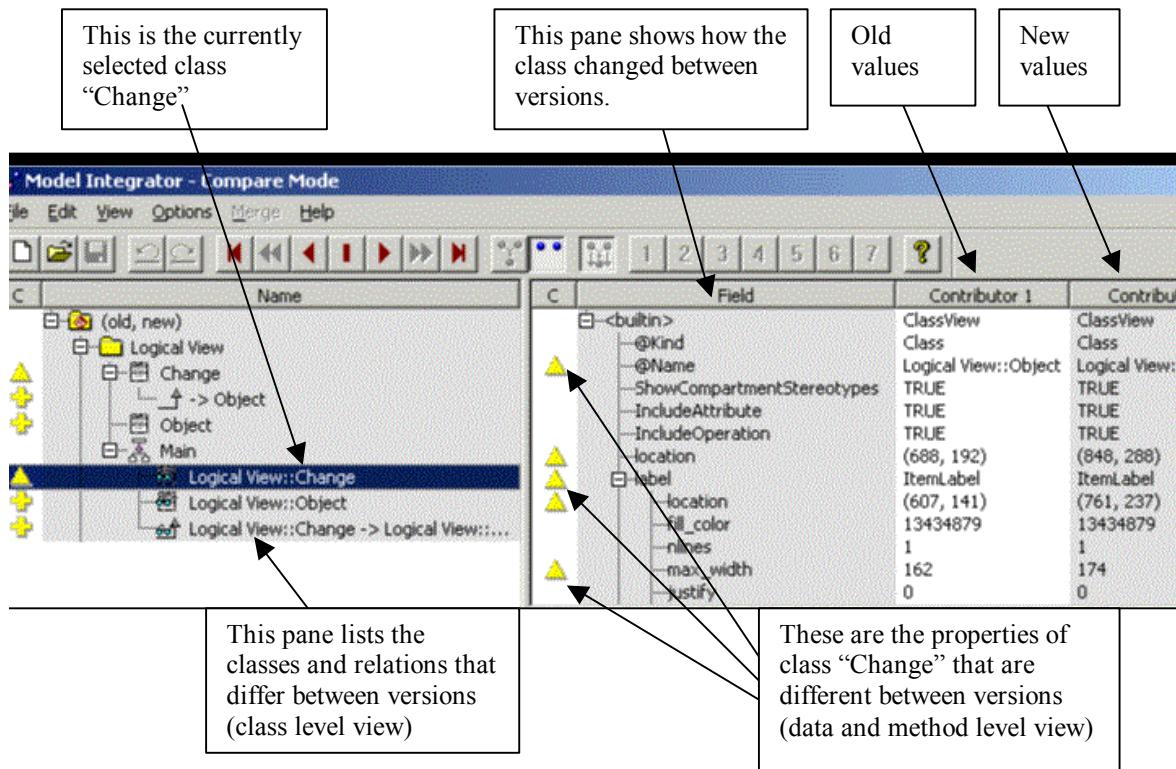
Figure 2.11: The modification of a picture illustrated by showing the modified version (left) and the original version (right with a red strike-through) in Word (Microsoft 1983).

An example of a popular, specialized 2D graphical application is Rational Rose (Rational Software Corporation 1991), a UML editor used in the specification and design of software systems (Figure 2.12a). The typical working view of classes in Rational Rose is that of a UML diagram where classes can be created and edited. Although there are several aspects to these classes that aren't represented using this graphical view, the advantage to having a diagrammatic representation is that it is easy to spot the relations between classes. Changes made to this diagram, however, are not shown in this view: instead a mostly text oriented, Model Integrator view provides change information in a separate window (Figure 2.12b). Its purpose is to help software engineers determine the differences between two branch versions of the project and to merge these branches rather than for change tracking.

Figure 2.12 illustrates a typical scenario of its use. A developer returns to the classes in Figure 2.12a some time after having last worked on it. In the interim, others have made changes to this model, and now the developer wishes to determine which classes have changed and how they have changed. Unfortunately, the working view presented in Figure 2.12a does not give any information regarding these changes, and so instead, the designer must access the Model Integrator in Figure 2.12b, which looks entirely different from the working view. It has a pane on the left hand side that displays the classes in the model as a tree, which differs considerably from the working view's graphical depiction. Although it can be argued that UML diagrams contain information that isn't visually represented (nor is there a need for it to be represented visually) some information is represented *best* in a visual form (Larkin and Simon 1987). Moreover, by presenting the information about changes made to the UML diagrams in a non-diagrammatic form, some viewers may miss the context in which the changes were made.



(a) An example of a class diagram in the working view provided by Rational Rose (Rational Software Corporation 1991).



(b) The Model Integrator view provided by Rational Rose (Rational Software Corporation 1991).

Figure 2.12: The regular working view of UML diagrams (a) and the Model Integrator view (b) which is the only (indirect) support provided by Rational Rose (Rational Software Corporation 1991) to track changes.

Another possibility of viewing graphical changes is through a graphical history. At its simplest, one could perhaps track changes in conventional drawing systems by using the undo/redo feature to trace backwards and forwards through the history of modifications that have been made to a picture over time. This is hardly a practical solution if many different people have made many changes and besides, few conventional systems maintain undo/redo information between sessions.

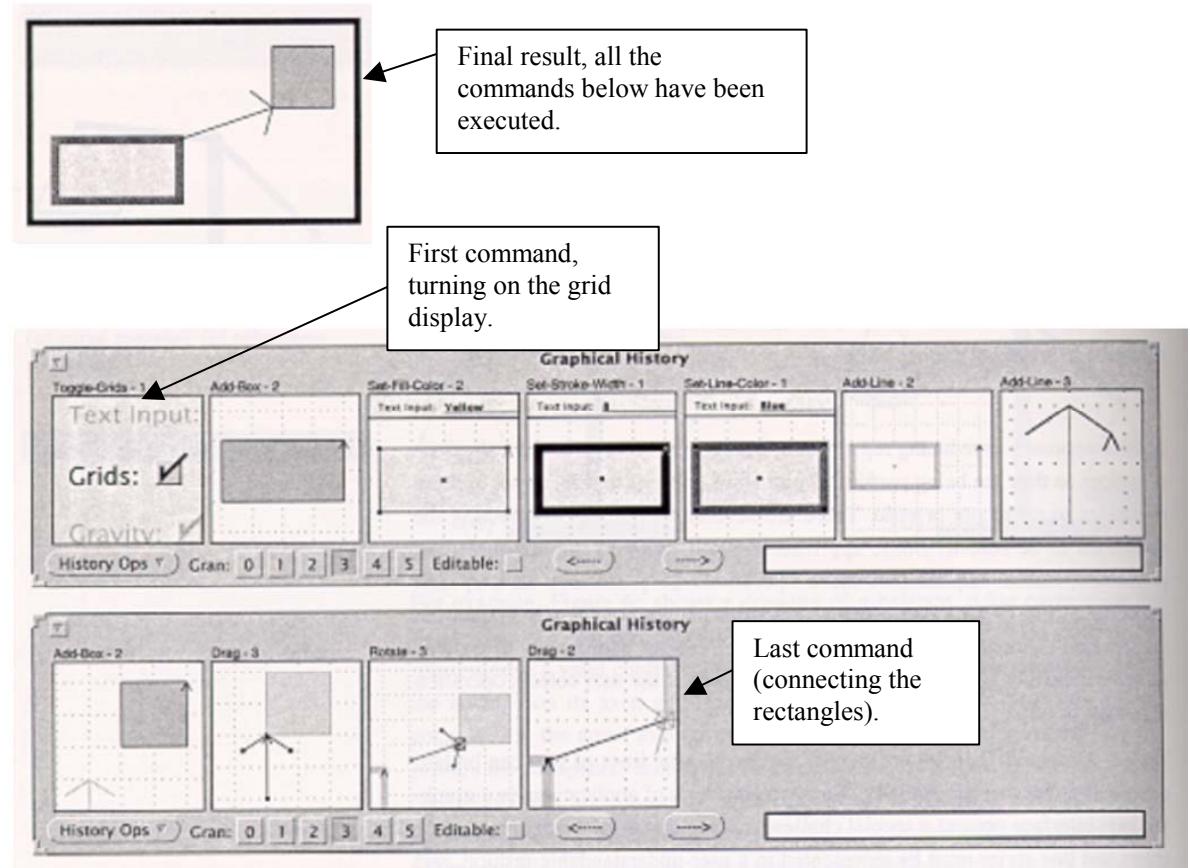


Figure 2.13: Editable graphical histories, showing the series of commands (bottom) needed to produce the final picture (top). Adapted from Kurlander and Feiner (1993).

A better example of how change can be supported via a graphical history is hinted at by Kurlander and Feiner's (1991; 1993) Chimera system. Chimera gives users the ability to edit a graphical history, and Figure 2.13 shows an example. The top picture shows the final completed picture as drawn by the user. The bottom picture consists of a series of

storyboard panels that show the sequential progression of that person’s drawing actions and their visual results. The system does not provide a panel for every action; rather, it determines when a change is significant enough to warrant its display. While this system was not developed for change awareness, the “storyboarding” technique it uses can potentially communicate change information in 2D graphical systems. I will elaborate further on how this technique can be used to represent changes in Chapter 4.

2.4 Summary

In this chapter, I briefly surveyed approaches to the management and display of change information. I started by describing the early version control systems such RCS (Tichy 1991), where changes between versions were displayed at a rather coarse granularity. I then explained how Configuration Management systems are better at tracking and managing changes in projects that involved multiple developers working on multiple files. After this I described how many of these early systems display changes in the form of a linear ‘differencing’ list, usually as a hard to decipher report presented separately from the changed files. Also, these systems can only handle ASCII text files, and could (at best) indicate if versions of binary files differed.

As text editors became increasingly sophisticated, better change tracking systems were developed to help authors understand the edits made to shared documents. Some systems embedded changes inline within the document as markups e.g., visual indicators like lines, highlighting, typeface changes and color differences. Other systems provide an overview of changes to indicate which parts of the document had changed, but do not show how those parts changed. Some systems do both (e.g. Seesoft), providing an overview that can be used to quickly pinpoint where changes were made while giving users the opportunity to dive into detail to see what has actually changed.

Considering how widespread two-dimensional graphical applications are, there is surprisingly little change awareness support for these types of systems. For example, there is no explicit support of change awareness built into the widely used UML editor

Rational Rose (Rational Software Corporation 1991). While the Model Integrator function can be pressed into service, as I indicated earlier, this function was meant for an entirely different task. Though not originally intended to provide change awareness information, the storyboarding approach employed in the Chimera system (Kurlander and Feiner 1991; 1993) for editing graphical histories is promising.

In the remainder of this thesis, I will draw upon some of the lessons learned from previous efforts to support change in text-based systems (such as the value of information filtering) and apply them to 2D graphical systems. In addition, I will return to some of the more promising mechanisms, such as overviews and storyboarding.

Chapter 3

Foundations of change awareness

As I described previously in Chapter 1, one of my research goals is to produce a conceptual framework for understanding change awareness in 2D graphical and collaborative workspaces, where multiple collaborators interact over the space at different times.

The prerequisite to understanding change awareness is determining the information necessary if people are to comprehend change in a collaborative workspace. These *informational elements* of knowledge for change awareness articulate, categorize and explain what information should be tracked and captured by an application as a change occurs and how this information could be useful to an end user. Once the informational elements have been specified, the next step is to display this information in a meaningful and useful fashion. Both are critical if we are to influence how a designer implements an effective change awareness system. Useful change information is of little value if it cannot be represented to the user in an effective way. Similarly, an excellent representation/visualization technique is of little use if it displays the wrong information. The details of the first step - the informational elements - will be discussed in this chapter; the key concepts of the second step – information display - will be described in Chapter 4.

This chapter unfolds by first listing five major components of awareness and then by summarizing how one of these components, Gutwin's (1997) *framework for workspace awareness*, forms the theoretical underpinnings of my framework. As will be discussed, Gutwin focused on real time (synchronous) groupware environments, where he was concerned with people's continuous maintenance of awareness of others in a visual workspace and how others were interacting with the space. In the next section, we will

see that many of the elements Gutwin identified for workspace awareness can be applied, extended and elaborated in a modified form to create a change awareness framework for different-time *asynchronous* work. In this situation, a person has been away from the workspace for a period of time and must be brought up-to-date on what has changed in the interim. While Gutwin does mention elements for maintaining awareness of changes in his framework, he did not elaborate on them: consequently, I will do that in this chapter.

3.1 Components of awareness

Awareness is commonly described as “knowing what is going on” (Endsley 1995: pp. 36). Human Factors and CSCW researchers have identified and explored five main categories of awareness that a person may maintain: situation awareness, informal awareness, conversational awareness, structural awareness and workspace awareness.

- 1) *Situation awareness* concerns people in critical environments, where failure to maintain a minimum level of awareness may result in undesirable consequences. Examples include aircraft piloting, and people in control and operating rooms. Awareness is dependent upon perceiving information in the surrounding environment, interpreting this information so that it is meaningful for the task at hand and, based upon this information, being able to predict what is likely to occur in the near future (Endsley 1995).
- 2) *Informal awareness* facilitates casual interaction and includes things such as knowing who is around in the workplace, what others are doing and whether or not they are available for conversation (Dourish and Bly 1992; Gutwin, Stark and Greenberg 1995).
- 3) *Conversational awareness* supports how people talk to one another in real time. Awareness is based on the visual and audio cues that one picks up when conversing with others, and helps one track how the conversation is going (Clark and Brennan 1991).

- 4) *Structural awareness* is concerned with social and task structures and therefore includes knowing how a work group is organized and the relationships between the people within it (Gutwin 1997).
- 5) *Workspace awareness* supports a group's real-time interactions over a visual workspace, where people use awareness to maintain their up-to-the-moment understanding of what is going on (Gutwin 1997).

Most of these categories of awareness are related to situations where a person is more or less maintaining continually their level of awareness to understand the current situation and its moment-by-moment changes. This differs substantially from the discrete nature of how a person comes to understand changes between asynchronous workspace interactions. However, we can relate Gutwin's framework for workspace awareness with a framework for change awareness, for both emphasize the visual workspace. In the remainder of this section, I will transform Gutwin's framework into a theoretical structure for change awareness. After providing more background on workspace awareness, I will expand on how its elements relate to change awareness.

3.2 Gutwin's notion of workspace awareness

Gutwin described a number of important characteristics of workspace awareness, or WA, as listed below (1997).

- 1) WA is constrained to the specific setting surrounding a visual workspace (or work surface), where people's actions and interactions are shaped by the nature of these workspaces. This includes how people are spatially located in and around the workspace, how they are conversing over it, and how they physically perform their actions within it. For example, when people enter the shared workspace, information about their location relative to the workspace is a component of awareness.
- 2) WA concerns information pertaining to a changing environment. This is why WA is defined as how people maintain their "up-to-the-moment understanding" of what is going on.

- 3) WA includes one's own interactions within the workspace, the current state of the artifacts within the space, and how others are interacting within the space. Thus it explicitly includes collaboration. As described by Gutwin (1997) "When a single operator flies a plane, for example, their awareness concerns the aircraft alone. When a piloting team is in charge, however, they have to know about the workspace (the cockpit), about the actions of other pilots, and how those actions relate to the workspace, in order to understand what the others are doing" (pp. 33).
- 4) People maintain this awareness from perceptual information gathered from the environment.
- 5) Maintaining awareness, while important, is secondary to some primary activity that the group is undertaking. That is, WA supports a primary activity but is rarely the primary activity in and of itself.

The entire collection of information that could be used for maintaining awareness is immense, although we expect that only a small subset of this information would be relevant to a person's needs at a particular moment. Except for very well defined situations, it may be impossible to describe *a priori* exactly what these awareness information needs would be during any given moment. However, Gutwin believes that most awareness needs stem from a common set of questions the answers to which provide the basic awareness information that likely apply to many situations. Gutwin (1997) suggests that these questions fall into the familiar "who, what, where, when and how" categories. The particular answers to these questions provide the *elements of knowledge for awareness*, and are the types of information that should be considered first by designers of real-time groupware systems.

Gutwin (1997) further distinguished two main categories of elements: the first is related to events, *as they are occurring* "elements of WA relating to the present"; the second is related to events that *have already taken place* "elements of WA relating to the past" (Gutwin 1997: pp. 37 – 39). In the next subsection I will summarize the first type of element (WA in the present) and in the sections that follow I will discuss the second type of element (WA in the past) and how it relates to change awareness.

3.2.1 Workspace awareness in the present

This collection of awareness elements consists of the information that is needed to track events in the workspace as they occur. Table 3.1 shows Gutwin's elements of knowledge contained within the “who, what and where” category of questions that may be asked about workspace events in the present. Gutwin did not include a discussion of the when and how categories.

For each of these categories of questions, there is a unique set of *informational elements* that provides answers to those questions. These informational elements are the specific pieces of information that a person would require in order to keep up with events as they occur in a collaborative and real time workspace.

Category	Element	Specific questions
Who	<ul style="list-style-type: none"> • Presence • Identity • Authorship 	<ul style="list-style-type: none"> • Is anyone in the workspace? • Who is participating? • Who is that? • Who is doing that?
What	<ul style="list-style-type: none"> • Action • Intention • Artifact 	<ul style="list-style-type: none"> • What are they doing? • What goal is that action part of? • What object are they working on?
Where	<ul style="list-style-type: none"> • Location • Gaze • View • Reach 	<ul style="list-style-type: none"> • Where are they working? • Where are they looking? • Where can they see? • Where can they reach?

Table 3.1: Elements of WA relating to the present from Gutwin (1997).

For example, knowledge of the ‘who’ category simply means that you know how many people are present in the workspace (if any), their identity, as well as being able to attribute a specific person to each action that is taking place (Table 3.1, second row). In the ‘what’ category (third row), awareness of action and intention means that you know what a person is doing both at a rudimentary level (e.g., typing some text) and at a more abstract level (e.g., creating a title). Awareness of artifact is knowing what object that another person is currently working on. Location, gaze, view and reach are all inter-

related in the ‘where’ category (fourth row): location refers to the part of the workspace where a person is currently working; gaze is the part of the workspace where a person is currently looking at; view is where they can potentially be looking i.e., within their field of vision, and reach includes the parts of the workspace that this person can potentially change (Gutwin 1997).

3.2.2 Workspace awareness in the past

Gutwin’s second collection of awareness elements consists of the information that is required for people to catch up with events that have already taken place in the workspace. Table 3.2 lists the elements of knowledge contained within the “who, what, where, when and how” categories of questions that may be asked workspace events in the past.

Category	Element	Specific questions
How	<ul style="list-style-type: none"> • Action history • Artifact history 	<ul style="list-style-type: none"> • How did that operation happen? • How did this artifact come to be in this state?
When	<ul style="list-style-type: none"> • Event history 	<ul style="list-style-type: none"> • When did that event happen?
Who (past)	<ul style="list-style-type: none"> • Presence history 	<ul style="list-style-type: none"> • Who was here, and when?
Where (past)	<ul style="list-style-type: none"> • Location history 	<ul style="list-style-type: none"> • Where has a person been?
What (past)	<ul style="list-style-type: none"> • Action history 	<ul style="list-style-type: none"> • What has a person been doing?

Table 3.2: Elements of WA relating to the past from Gutwin (1997).

Determining ‘how’ the workspace has changed involves two elements: *action history* and *artifact history* (second row, Table 3.2). Action history describes the unfolding of events that changed the workspace. Artifact history includes details about the process of how an object was changed over time (Gutwin 1997). However, there is nothing in Gutwin’s framework that specifically describes how the workspace itself has evolved over time.

Information about ‘when’ something has occurred (third row) is described by the *event history* of the workspace. This element indicates the time at which things occurred in the workspace. ‘Who’ provides a *presence history* of people in workspace (Gutwin

1997), that is, of knowing who has visited a particular location and when this visit occurred (fourth row).

Although there are potentially many aspects to the ‘where’ category of questions e.g., where did events take place, where have artifacts been etc., Gutwin mentions only the *location history* of other people, which indicates where a person has been in the workspace (1997). Finally the ‘what’ category of questions lists the *action history* of a person, which describes what actions have another person engaged in (last row).

Gutwin’s framework collects WA elements that relate to the past, which provides a good foundation for the creation of a framework for change awareness. However, he was mostly concerned with elements relating to the present – thus he did not elaborate on past elements beyond this initial list.

Consequently, I found it necessary to adapt and expand upon the base structure that he laid out for a number of reasons. First, there are additional basic informational elements about the past that were not described by Gutwin. Second, as I have already mentioned Gutwin’s framework dealt with real time workspace events. Consequently he did not focus on representing workspace information that accumulates over time (changes in the workspace). Finally, different people can perceive the workspace in different ways. This includes an *artifact-based view*, a *person-based view* and a *workspace-based view*. I will expand further on the first and third issues in the next section. Strategies for handling the second issue will be deferred to Chapter 4 when I talk about the representation of change awareness information. From this point onwards in this thesis, I shift the focus from Gutwin’s framework for workspace awareness and begin to outline and describe in detail my own framework for change awareness.

3.3 Informational elements for change awareness

When trying to catch up with changes the first piece of information that a person needs to know is “Is anything different since I last looked at the work?” Obviously a change awareness system must provide the answer to this question in a very lightweight fashion.

Afterwards, the person can then probe for further details by trying to find out the specifics of a change. The specific information that a person may require in order to track changes will vary from situation to situation. It will depend upon the task that is being performed, the person who is carrying out the task, as well as the surrounding environment. But in a fashion that is similar to the scenario covered by Gutwin's framework for workspace awareness, it is possible to describe at a high level the questions that may be asked. This set of questions includes:

- 1) Where have changes been made?
- 2) Who has made the changes?
- 3) What changes were made?
- 4) How were things changed?
- 5) When did the changes take place?
- 6) Why were the changes made?

However, a change awareness framework must account for the fact that people may need to view aspects of the workspace in different ways at different times. In particular, a person may query the workspace for changes in terms of:

- The artifacts that exist within it (*artifact-based view*),
- The people who work within it (*person-based view*),
- Or the workspace may be viewed as one locale or as a collection of related locales (*workspace-based view*) where a person is interested in the changes and events that have taken place in one or more locales.

The specific perspective that a person has of the workspace will have an impact on the information that he or she is interested in and the way that the information is requested and represented. In terms of the artifact-based view, the person will be interested in changes made as they relate to particular workspace artifacts, and will make various queries about those changes. Examples include: how has this item changed, and what has been done to this item? From the person-based view, an individual wishes to know about the changes that were made by another collaborator. Queries about changes

will therefore be focused on this person e.g., what did he do while I was away? On the other hand, someone with a workspace-based view would be interested in and inquiring about the events that have taken place in a specific location e.g., what changes and events took place in this space? This location can consist either of the workspace as a whole or portions of the space that are somehow logically related e.g. a specific room in a room-based groupware system or a particular spatial region.

Of course there is a strong relation between the three workspace perspectives and the six categories of awareness questions. An individual that holds a person-based perspective will focus heavily on the ‘who’ category of questions. Someone that holds an artifact-based view of the workspace may focus instead on the ‘what’ category of questions and try to determine what changes were made to specific objects. Yet another person that holds a workspace-based view of the workspace may focus on the ‘where’ category of questions. Alternatively the person with a workspace-based view may focus on ‘what’ was done in the part of project that he or she holds an interest in. The main point is that the person’s particular view of the workspace will influence the value that he or she attaches to each category of question. As will be seen, however, the specific example questions that are unique to each of the six high level categories awareness questions can be asked from any of the three workspace perspectives.

The following subsections will describe in detail the informational elements associated with each category of question as well providing some specific example questions that a person might ask about changes.

3.3.1 Where?

Location in a 2D graphical workspace could be a simple Cartesian spatial region, or a direct digital analogue of physical demarcations, e.g. the rooms in a room-based system such as TeamRooms (Roseman 1996; Roseman and Greenberg 1996a; Roseman and Greenberg 1996b) or locations may be more abstract in relating workspace entities to each other, e.g., the different logical or conceptual parts of a collaborative project. In all cases, the location of a change provides valuable clues regarding its context, which in

turn guides people towards further exploration. For example, a person may ask if a given change was part of the project component that is undergoing extensive rework, that is, the change occurred in the same place where many other changes are also occurring.

Table 3.3 shows the specific questions that may be asked to learn ‘where’ changes have occurred with respect to each of the three workspace perspectives. As I already mentioned, the difference is how queries about changes made to the workspace are formulated within each perspective. With the artifact-based view, the questions could be asked in terms of a specific object in the workspace. Where is it now? Where was it before? Where has it been since I have been away? From the person-based view, the example questions may be asked about a specific collaborator. Where has this person visited or looked in the workspace? Where did this person change things? From the workspace-based view, the questions asked would inquire about the different events that have taken place in the space since a person has been away. Where in the workspace have people visited? Where were the artifacts moved?

The informational elements that will answer the ‘where’ category of questions include Gutwin’s location history (described in section 3.2.2). Also I add gaze history and edit history. Location history refers to the parts of the workspace that have been visited by a person. Gaze history includes the parts of the workspace that a person has looked at. The difference between location and gaze history is that while a person may have been present in a general location, he or she may not have actively attended to everything that went on there. Although location and gaze history do not directly provide information about changes that have been made, they do indicate the parts of the workspace that have been visited or viewed and the frequency of these visits. This provides strong clues as to where one should look for changes. Edit history, on the other hand, explicitly deals with the changes that were made. Awareness of ‘where’ edits occurred is vital to routine project management as it provides strong clues as to the progress that has made towards satisfying project-level goals. By this very fact, the location of changes provides strong cues to the answers to other “who, what, why, and how” category of questions.

Where			
Informational Elements	Specific questions		
	Artifact based view	Person based view	Workspace view
<ul style="list-style-type: none"> • Location history • Gaze history • Edit history 	<ul style="list-style-type: none"> • Where was this artifact (when I left)? • Where is the artifact now? • Where has this artifact been during the time that I have been away? 	<ul style="list-style-type: none"> • Where in the workspace has a person visited? • Where in the workspace has a person looked at? • Where in the workspace has a person made changes? 	<ul style="list-style-type: none"> • Where have people been in the workspace? • Where were artifacts in the workspace? • Which parts of the workspace have people looked at? • Which parts of the workspace have people made changes in?

Table 3.3: Informational elements and specific workspace questions related to the ‘where’ aspect of change awareness.

Also a person tracking changes will better comprehend the person-based view of where changes occur by integrating the answers to the questions listed in Table 3.3 with structural awareness cues. Structural awareness, if you recall from Section 3.1, is an understanding of how people in a project are organized, and this complements change awareness well. Structural awareness helps one narrow down where in a workspace to look for interesting changes (e.g., the ones made by a specific person) and it can be used to predict who made particular changes (e.g., those within a specific part of the workspace), helping to narrow down whom to contact regarding a particular change.

3.3.2 Who?

Answers to these questions are important for several reasons. In collaborative environments, knowing who made a change becomes an opportunity to query that person directly for further information. Also, people may attend to changes differently depending upon who made them. For example Neuwirth, Chandhok, Kaufer, Erion, Morris and Miller (1992), described how collaborators could be less interested in seeing

changes made by co-authors that he or she has known for a long time and more interested in seeing changes made by less trustworthy co-authors.

In Table 3.4, I provide a more detailed breakdown of the ‘who’ questions from the different workspace views. To Gutwin’s (1997) concept of *presence history*, I add *identity*, *readership history* and *authorship history*. While presence history tracks if anyone was present, identity indicates the specific individual associated with a change or event. Establishing identity is needed to provide the context for answering the person-based view of workspace changes. For example, a team member may be responsible for auditing and upgrading different parts of the project. If his or her identity is associated with a particular change, it may better help other team members (who may prefer their original version) accept that change. Readership history carries identity even further by listing all the people who have viewed a particular artifact, or conversely, listing all the items that a particular person has seen. This is important as knowing that someone has viewed the changes without raising any objections or making any further changes could imply an implicit acceptance of the current work. By a similar vein, authorship history can list the people who have made changes to an artifact, or list all the artifacts that a particular person has changed. Tracking readership and authorship history can, for example, be used to gauge progress of the project through a process-oriented lifecycle. In such a case knowing who has seen an object and ‘signed off’ on it is an important part of workflow management and document routing.

Who			
Informational Elements	Specific questions		
	Artifact based view	Person based view	Workspace view
<ul style="list-style-type: none"> • Presence history • Identity • Readership history • Authorship history 	<ul style="list-style-type: none"> • Who has looked at this artifact? • Who has changed this artifact? 	<ul style="list-style-type: none"> • Who has this person interacted with? • Who made changes with this person? 	<ul style="list-style-type: none"> • Who has been in the workspace? • Who has looked at the workspace? • Who has made changes to the workspace?

Table 3.4: Informational elements and specific workspace questions related to the ‘who’ aspect of change awareness.

3.3.3 What?

As shown in Table 3.5, the ‘what’ category of questions leads to answers that produce a picture of the *action history* of the workspace. Gutwin (1997) described the two ways that action history can answer these questions. First, it can be used to track all of the low level actions that a person has done, e.g., creating, labeling and positioning a circle in a diagram. Knowledge about actions that people have engaged in while one was away is important. When people are asked to describe what is new in the workplace it is frequently in terms of the actions and events that have taken place. Sometimes there is, of course, a need to put all of these lower level actions in the context of the higher goals. So Gutwin also described action history from a higher-level perspective of the low level changes in a way that considers the goals that motivated these actions, e.g., the labeled circle was created in order to represent a new person in an organizational chart.

However, the questions and corresponding answers presented in Table 3.5 are often the only information that a programmer will have when he or she chooses to add change awareness support to an application. The problem is that it is difficult to ascertain and store the motives behind a series of low level changes. One could use spatial proximity (i.e., changes that are located near to each other) or temporal proximity (i.e. changes that occur at about the same time) as predictors of inter-relatedness, but often these methods will fail because the sub-steps for achieving several different high level goals may often be interleaved. Thus, I will postpone discussing the higher-order view of changes until Section 3.3.6, and instead focus here on only the significance of low-level changes. It is important to point out, though, that people are able to relate and combine several low-level actions to derive a higher-level action if they are given enough contextual information to understand the relationships between the lower-lever actions.

What			
Informational Elements	Specific questions		
	Artifact based view	Person based view	Workspace view
• Action history	• What changes have been made to the artifact?	• What artifacts has a person looked at? • What artifacts has a person changed? • What activities has a person engaged in?	• What changes have occurred in the workspace? • What artifacts were viewed? • What artifacts were changed?

Table 3.5: Informational elements and specific workspace questions related to the ‘what’ aspect of change awareness.

The specific questions associated with the ‘what’ category varies depending upon the perspective that a person has of the workspace. These questions are shown in table 3.5. For the artifact-based view, inquiries are made about the changes that have been made to a particular artifact. From the person-based view, the questions ask about what actions has a person undertaken. With the workspace-based view, the questions ask about the actions that were undertaken within the workspace or actions that were carried out on the artifacts in the workspace.

3.3.4 How?

The ‘how’ category of questions asks how the current workspace differs from the way that it was before (Table 3.6). The answers to these questions can be integrated to derive one of two historical views of changes. The first is in terms of the *process history* of the workspace, which indicates incrementally how the workspace evolved from its previous state (i.e., the state that it was in when one last looked at it) to its present state. This is useful when a person is interested in the mechanical means (the intermediary steps) that produced a change or group of changes as well as the end result. Thus process history is tightly coupled with action history. The difference is that the action history consists of all the actions that have occurred while one was away, while process history relates and abstracts a subset of all actions into a series of steps in a process. The process view is

important for, as Gutwin (1997) described, people may have trouble interpreting instantaneous changes. Thus describing all the sub-steps involved in a change may help to clarify what happened. Also, the process-oriented view describes the *evolutionary context* of changes (i.e., the specific details of the circumstances faced by the person who made the change at the time that the change occurred), and thus can provide valuable insight on ‘how’ and ‘why’ things came to be in their present state.

How			
Informational Elements	Specific questions		
	Artifact based view	Person based view	Workspace view
<ul style="list-style-type: none"> • Process history • Outcome history 	<ul style="list-style-type: none"> • How has this artifact changed? 	<ul style="list-style-type: none"> • How has a person changed things? 	<ul style="list-style-type: none"> • How has this workspace changed?

Table 3.6: Informational elements and specific workspace questions related to the ‘how’ aspect of change awareness.

Of course, a person may only be interested in the final result. This is the second historical view of ‘how’ a workspace has changed, the *outcome history*. The outcome history presents only a ‘bottom line’ understanding of a change where it highlights only those things that differ between the initial and the final state.

The choice of process *vs.* outcome history will depend largely upon the task at hand. For example, a graphic artist may be interested in the technique used to produce some visual effect. In this case, this person will want to see (and thus learn) the process history of the workspace. On the other hand, a newspaper editor reviewing an article submitted by a reporter is far too busy to be concerned with the rough drafts produced by this person, and would thus be interested only in the outcome history of the article. Consequently, it is important that software support for change awareness provide the ability to discover both the process and outcome history of a workspace.

3.3.5 When?

The timing and ordinality (sequential order) of changes is revealed by the answers to the questions of ‘when’ changes occurred as listed in Table 3.7. When a change occurred, particularly if it overrides an earlier change by another person, is often of great significance and affects the perceived importance of a change. For example, a person may only be interested in recent workspace events, or a person may only be interested in changes that occurred within a specific period of time.

When			
Informational Elements	Specific questions		
	Artifact based view	Person based view	Workspace view
• Event history	<ul style="list-style-type: none"> • When was this artifact changed? • When was a particular change to this artifact made? • In what order were changes made to this artifact? 	<ul style="list-style-type: none"> • When did a person make changes? • When did a person make a particular change? • In what order did this person make changes? 	<ul style="list-style-type: none"> • When were changes made to the workspace? • When did a particular change in the workspace occur? • In what order did changes to the workspace occur?

Table 3.7: Informational elements and specific workspace questions related to the ‘when’ aspect of change awareness.

The timing and ordinality of changes constitute the *event history* of the workspace, and it provides the *chronological context* for understanding and interpreting changes giving clues to the ‘where’, ‘who’, ‘what’, ‘how’ and ‘why’ categories of questions.

3.3.6 Why?

Knowing the thought and motives behind a change can be important for accepting the changes that others have made (Cross 1990). The questions that a person will ask to discover ‘why’ changes were made are summarized in Table 3.8.

Why			
Informational Elements	Specific questions		
	Artifact based view	Person based view	Workspace view
• Cognitive history • Motivational history	• Why was this artifact changed?	• Why did a person make that change?	• Why was that change made in the workspace?

Table 3.8: Informational elements and specific workspace questions related to the ‘why’ aspect of change awareness.

A historical view of ‘why’ changes were made includes both the *cognitive history* and the *motivational history*. Cognitive history describes the logic or reasoning that may be behind a change, which is a rational reconstruction of the person’s goals and plans. Motivational history deals more with the impulses or desires that are the impetus for making a change, which is the actual reason why a person did something at a moment in time. The reason that they are separate elements is because a change may be based upon a well thought out and carefully conceived plan or it may be more of a spur of the moment thing as one reacts to the current situation (Suchman 1987). Also, some changes are completely unintended accidents.

Although it is not always needed, knowing ‘why’ a change was made is obviously an important step for coming to understand and accept it. For lower-level changes that are the parts of a grander higher-level change, the motivating factors may be painfully obvious. In this way providing the motivational history for simple changes may be too effortful (and distracting from the main task) to explain. Also, describing all the motives behind a change and detailing all the reasoning behind each event is extremely difficult for computers to do automatically. This is because it often draws upon a person’s accumulated technical expertise and implicit or ‘hidden’ cultural information relating to group priorities, work practices, and short and long term goals. Today’s computing systems lack the ability to sense these technical and cultural factors motivating a change and the intelligence needed to produce a truly comprehensive picture of the cognitive history of the workspace. Consequently, most ‘why’ information will likely be generated explicitly by authors e.g., as annotations to changes or as design rationales.

3.3.7 Discussion

In this section I have described in detail the information that can be used by a person to track changes in a collaborative project. The informational elements were classified according to the categories of change awareness questions. These categories are inter-related and inter-dependent. When a person is tracking changes he or she may start with the highest-level question, ‘Has anything changed?’ From that point the person will make inquiries about changes from one or more of particular perspectives - artifact, person, or workspace-based - that make the most sense to him or her. The inquiries can be directed towards a specific collaborator, ‘What has this person done?’ Alternatively, the process of inquiry can take the form of an examination of a particular artifact, ‘How does this object differ from before?’ or it can take the form of an inspection of a select portion of the workspace, e.g., ‘What has happened in this corner of the project?’

Within the bounds of the chosen perspective, a person can ask specific questions from these categories to probe for further details of changes. The specific category that a person begins with (where, who, what, how, when or why) is not fixed. As mentioned previously, it will be influenced by the workspace perspective that is held e.g., if a person is making inquiries from a person-based view then the ‘who’ category may be of the most pressing urgency. Also, as I have shown in the previous sections, queries made in one category of question are not isolated from the other categories. The process of inquiry can occur in parallel as someone delves for answers in more than one category at once. For example, take the case of a project manager who is reluctant to have certain parts of the project undergo anymore changes, or who has severe misgivings about the work of specific team members. When the manager returns to the project after a period of absence and discovers that many changes have been made, he may immediately try to determine exactly where changes were made and specifically who made those changes.

The answers to the questions from one category of question may also inspire additional inquiries in another category. For example, a person who is tracking the historical context of the changes to a software system may start by asking about ‘when’

most of the changes occurred. Upon discovering this information she notes that the code was most volatile during the period when the system was being ported from one operating system to another. Since this person is interested in learning how to make these conversions herself, her queries immediately focus on the process history of the software as she tries to determine exactly what the programmers had to do in order to make the port.

Furthermore, the answers to the questions that a person asks about one category of question may directly provide him with further information. For example when a person knows exactly where changes occurred, she then knows who made those changes (because she knows who is responsible for which portions of the project). Or the person may be able to make predictions about the answers to the other categories of questions based upon the information that she gets from one category. For example, when a person learns that a specific team member made a change, she can make guesses as to how the change was made. These guesses are based upon her personal knowledge of the person who made the changes and the techniques that he has employed in the past.

Although a single answer may provide information about multiple categories of questions, the main point of the framework is to ensure that designers of change awareness systems actually capture this change information, so it can later provide the person who is tracking changes with the information needed to answer these questions.

3.4 Summarizing the components of awareness

In this chapter, I have showed how the area of change awareness fits within the previous research that was conducted into awareness in general. Specifically Gutwin's (1997) framework for Workspace Awareness for real time interaction over a visual work surface provides elements that I used as a foundation for a framework for understanding asynchronous change. I argued that the "elements of knowledge" which answers the basic "who, what, where, when, how, and why" questions are also important when trying to familiarize oneself on the changes made since one was away from the workspace.

However, a few adaptations and extensions of Gutwin's (1997) framework were necessary. I added an additional category of change awareness questions for 'where' changes took place. This was needed because the information provided by this category not only helps place the changes in context of workspace locations but knowing the place where events took place can also provide cues as to where one can probe for further details about changes. Additional informational elements were also needed to fill in information missing from some of the other categories. Finally because different people may see the workspace from different perspectives (person-based, artifact-based, workspace-based), I added to the framework the ability to offer change awareness information in different ways. I also noted that queries made about one category of question do not operate in isolation from the other categories. Queries may occur in parallel, or answers derived from one category may inspire additional queries in another category or may allow the prediction of the answers to the other categories.

Table 3.9 collects in one place all of the high-level categories of change awareness questions that may be asked, the informational elements that answer each of these questions, as well as examples of the specific questions that may be asked from a particular perspective.

Where			
Informational Elements	Specific questions		
	Artifact-based view	Person-based view	Workspace-view
<ul style="list-style-type: none"> • Location history • Gaze history • Edit history 	<ul style="list-style-type: none"> • Where was this artifact (when I left)? • Where is the artifact now? • Where has this artifact been (during the time that I have been away)? 	<ul style="list-style-type: none"> • Which parts of the workspace has a person visited? • Which parts of the workspace has a person looked at? • Which part of the workspace has a person made changes? 	<ul style="list-style-type: none"> • Where have people been in the workspace? • Where were artifacts in the workspace? • Which parts of the workspace have people looked at? • Which parts of the workspace have people made changes in?
Who			
<ul style="list-style-type: none"> • Presence history • Identity • Readership history • Authorship history 	<ul style="list-style-type: none"> • Who has looked at this artifact? • Who has changed this artifact? 	<ul style="list-style-type: none"> • Who has this person interacted with? • Who made changes with this person? 	<ul style="list-style-type: none"> • Who has been in the workspace? • Who has looked at the workspace? • Who has made changes to the workspace?
What			
<ul style="list-style-type: none"> • Action history 	<ul style="list-style-type: none"> • What changes have been made to the artifact? 	<ul style="list-style-type: none"> • What artifacts has a person looked at? • What artifacts has a person changed? • What activities has a person engaged in? 	<ul style="list-style-type: none"> • What changes have occurred in the workspace? • What artifacts were viewed? • What artifacts were changed?

Table 3.9: Summary of all the change-related categories of questions, informational elements associated with those categories and examples of specific questions that may be asked.

How			
Informational Elements	Specific questions		
	Artifact-based view	Person-based view	Workspace-based view
<ul style="list-style-type: none"> • Process history • Outcome history 	<ul style="list-style-type: none"> • How has this artifact changed? 	<ul style="list-style-type: none"> • How has a person changed things? 	<ul style="list-style-type: none"> • How has this workspace changed?
When			
<ul style="list-style-type: none"> • Event history 	<ul style="list-style-type: none"> • When was this artifact changed? • When was a particular change to this artifact made? • In what order were changes made to this artifact? 	<ul style="list-style-type: none"> • When did a person make changes? • When did a person make a particular change? • In what order did this person make changes? 	<ul style="list-style-type: none"> • When were changes made to the workspace? • When did a particular change in the workspace occur? • In what order did changes to the workspace occur?
Why			
<ul style="list-style-type: none"> • Motivation history • Cognitive history 	<ul style="list-style-type: none"> • Why was this artifact changed? 	<ul style="list-style-type: none"> • Why did a person make that change? 	<ul style="list-style-type: none"> • Why was that change made in the workspace?

Table 3.9 (continued): Summary of all the change-related categories of questions, informational elements associated with those categories and examples of specific questions that may be asked.

Chapter 4

Display of awareness information

In Chapter 3, I described the basic categories of questions that may be asked about changes in the workspace. Regardless of the importance of this information, it may be useless if it cannot be presented and communicated in an effective manner. Consequently, in this chapter I focus on some important concepts related to the display of this information.

In the first section, I describe the dimensions for displaying workspace changes, which include: *presentation*, *placement* and *persistence*. Presentation describes how awareness information is portrayed, placement determines where the information is located (Gutwin 1997), while persistence indicates how long information about changes remains visible.

In the subsequent section, I discuss how displaying change awareness information must be done in a way that minimizes the *interpretation cost* of this information i.e., the amount of mental effort that must be expended to make sense of something (Gutwin 1997). We must also consider *acquisition costs*, the amount of effort expended to find or get a hold of the information. For example, requiring that a person run the application just to see if anything has changed is overkill for what should be a simple task.

Of interest are the actual methods used to display changes, and in the next section, I describe how Bertin's original set of *visual variables* (1967) as well as Carpendale's (2001) elaborated set can guide the display of changes. As defined by Bertin, visual variables are the different ways in which a mark on a piece of paper may be varied. I also include two other display mechanisms: Kurlander and Feiner's (1991; 1993) *storyboarding technique*, and *text*.

I conclude this chapter by discussing how change filtering can reduce onscreen clutter and thus help avoid overload. We will see that there are several types of filters and filtering strategies, where each has a different effect on what is displayed and how it appears.

4.1 Dimensions of change display

Gutwin (1997) provides two dimensions, placement and presentation, for portraying workspace awareness information (Figure 4.1, front pane). I believe these also apply to change information.

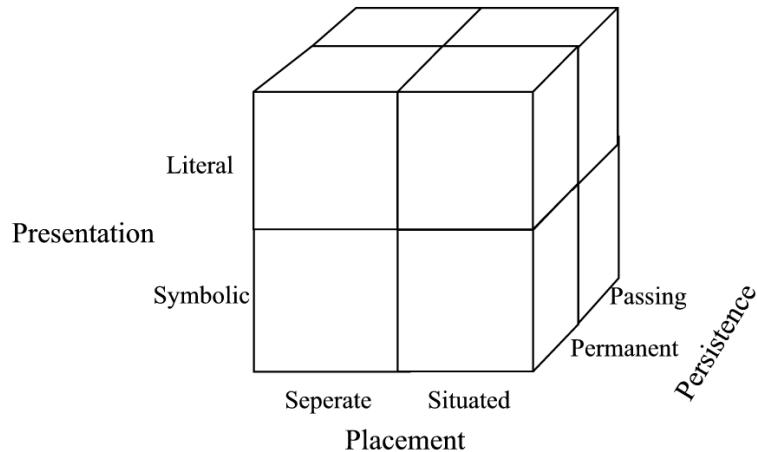


Figure 4.1: Presentation, placement and persistence of awareness display dimensions.

Placement determines where the awareness information is located in the display. Awareness information has *situated* placement if the information is located in the part of the workspace where the event occurred, and it has *separate* placement if is located somewhere else. Gutwin argues that situating awareness information takes advantage of a person's existing familiarity with the workspace, for it provides context. However, if many changes and events are taking place in the space over time then the potential downside is clutter leading to overload requiring increased effort to interpret the changes. Thus, some balance must be struck between context and overload.

The presentation dimension of Figure 4.1 classifies the display of awareness information as *literal* when it describes awareness information in the same form that it is gathered. In terms of change awareness this would mean that all the details about changes would be shown. It is *symbolic*, when only a subset of the information about a workspace event is displayed (Gutwin 1997). While a literal presentation may be easier to understand and interpret, in terms of change awareness, because of the potentially large amount of information that can accumulate as changes occur over time an overly literal presentation may sometimes be more a nuisance than a benefit. This was found to be the case in my own investigation of potential change display mechanisms summarized in Chapter 5 and published as Tam, McCaffrey, Maurer, and Greenberg (2000). During this study, many test participants expressed a desire for useful abstractions that combine rudimentary change information into one higher-level conceptual change. For example, one participant noted while watching the animated replay of a class name being shown, "...I don't need to see each and every character being typed just to see a name change!" Of course, care must be taken to make these abstractions understandable, e.g., by using already familiar representations or notations. This minimizes the cost of acquiring information while maximizing its benefits due to the added structure and organization.

Based upon my previous findings (to be discussed in Chapter 5), I add a third dimension, *persistence*, to Gutwin's classification. Persistence refers to how long the information is displayed (Figure 4.1 side pane). The display of information is *permanent* if it is always visible and *passing* if it only appears for a certain period. We noticed how study participants frequently complained when important information disappeared off the screen. Conversely, they also indicated that screen clutter might occur with the mechanisms that constantly displayed all changes. Thus, there's a need to classify change information according to how long it should stay visible.

With permanent persistence, the effort needed to find changes i.e., the acquisition cost is low because the information is always there. Ideally, a person merely has to shift their gaze over to see the information. Because people can become accustomed to the

occurrence of workspace events, they can also ignore things that do not interest them and pay closer attention to things that are of interest (Gutwin 1997).

With passing persistence, information about changes is presented only for a limited duration. This is useful when the information applies only to a specific portion of the project (artifact or group of artifacts) being viewed, or when the change information otherwise becomes irrelevant.

The matrix in Figure 4.1 suggests that these dimensions can be combined, giving eight possibilities. For example, a literal, situated and passing display of changes is depicted in Figure 4.2a. The figure shows an animation of a changed circle (by using a ‘replay’ technique) where the circle literally retraces the path that it took as it was moved. It is situated because the animation occurs in the same place that the change actually happened. The persistence is ‘passing’ because once an animation has replayed a change, the information is gone. Figure 4.2b shows two other examples within a concept map editor. The first illustrates the symbolic, situated and permanent octant, where color value (shades of gray) is used to indicate changed ‘Jim’ and ‘Jack’ nodes. Thus, it is symbolic because changes are mapped to a gray scale value, situated because the shading is applied directly to the node that was changed, and permanent because the color values are always on. Figure 4.2b also portrays an example of the symbolic, separate, and passing octant, where a person can raise a node’s change details in a pop-up as a text description by mousing-over the node. Thus it is somewhat separate as the information appears outside the changed node, it is symbolic as it uses the text to describe the changes, and passing because the pop-up disappears when the person moves the mouse off the node.

In summary, these three dimensions provide the designer with a means of classifying change information. I now turn to other display issues, where we need to represent the change information in an easily understood and readily accessible fashion.

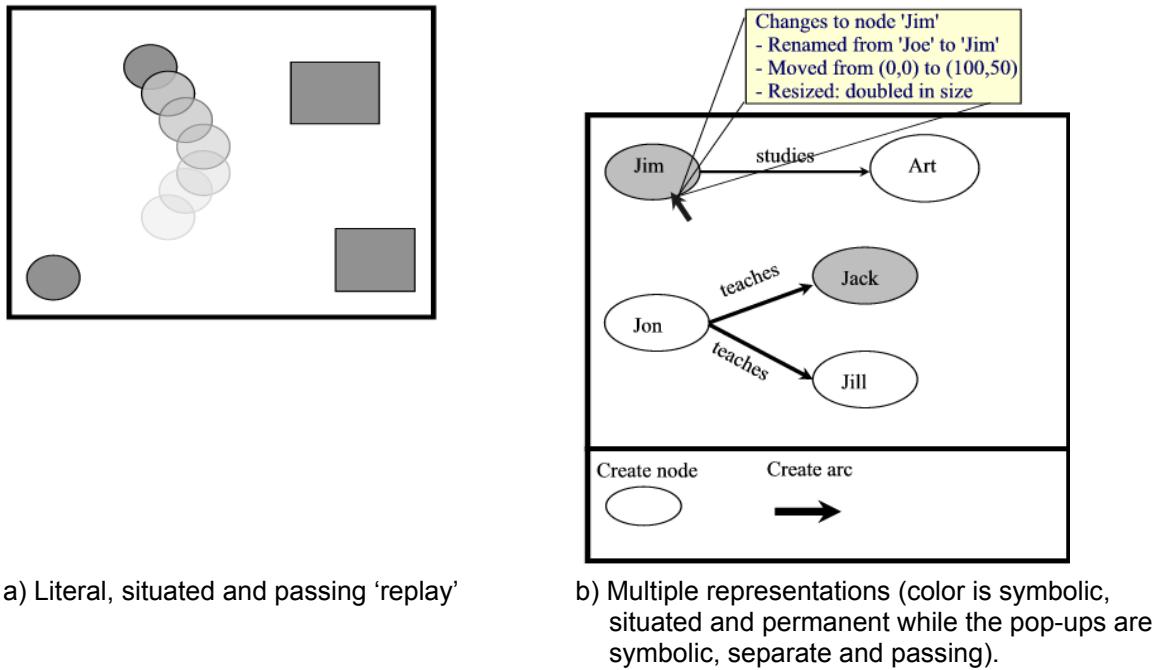


Figure 4.2: Example mock-ups illustrating several display dimensions.

4.2 Reducing interpretation and acquisition costs

The display of information is not a trivial matter. As stated by Endsley, “...the way in which information is presented via the operator interface will largely influence [awareness] by determining how much information can be acquired, how accurately it can be acquired, and to what degree it is compatible with the operator’s needs” (Endsley 1995: pp. 50). Care must be taken not to overload a person with so many details at once that he or she cannot make any sense of it. Visualizations such as overviews and details (Shneiderman 1996) or focus and context views (Furnas 1986) can be used as ways of representing a subset of the changes in a meaningful fashion. Also it is important that the mechanism employed by an individual to query for changes and the way the information is displayed makes sense to this person regardless of their perspective of the workspace (person, artifact or workspace-based as described in Chapter 3). Finally, if it requires too much effort for someone to find out what has changed then he or she may not bother.

4.2.1 Reducing acquisition and interpretation costs

One difficulty that can arise when interpreting a display of accumulated changes is the sheer volume of information. Because people are limited in the amount of information that they can actively process at a particular point in time (Card, MacKinlay and Shneiderman 1999) there is a real danger of overload. Thus, people need all the help they can get. I suggest that many information visualization techniques (Card, MacKinlay and Shneiderman 1999) developed for letting people explore and understand large data sets can also be applied to change awareness information display.

As mentioned in Section 4.1, Gutwin (1997) described the importance of placing the workspace events in relation to familiar workspace artifacts and landmarks as this provides temporal and spatial context. Yet this is problematic when workspaces are large and screen displays are small: there is a fundamental tradeoff between seeing the global context of a change (in the macroscopic view) as well as its detail (in the microscopic view). As I indicated in Chapter 2, there are many benefits to having an overview of a large workspace. The abstract river metaphor used in Theme River (Havre, Hetzler and Nowell 2000) or the literal high level overview depicted in the Seesoft system (Eick, Steffen and Sumner 1992) are effective visualizations for describing trends in large amounts of information. As we saw with case of diff (Hunt and McIlroy 1975), when only the microscopic view of changes is presented, trying to interpret the change information out of its global context can be difficult even if only a modest number of events have taken place.

Shneiderman's (1996) visualization mantra "overview first, zoom and filter, then details-on-demand" (pp. 337) suggests that change displays should first provide an overview of the project and its changes which lets people easily detect patterns or trends in the changed data and permits rapid searching for changes of interest. These changes can then be explored in detail (Card, MacKinlay and Shneiderman 1999). For example, in Section 2.2.3, I described how Word (Microsoft 1983) utilized color to represent change. If viewed from its zoomed overview, it shows potential areas of interest

(indicated by the color patches). The person can then progressively zoom into these areas of the document to examine them in detail. Seesoft provided a two-step view: the overview, and then a click to access the program code in full size. In contrast, with Theme River users had no way of getting from the overview to the actual details.

There are problems with the overview approach. As one zooms in for details the surrounding context can be lost (Schaffer, Zuo, Greenberg, Bartram, Dill, Dubs and Roseman 1996). This is certainly true in the case of Word's zooming feature, as it does not allow simultaneous display of the overview and detailed view. Also, because overviews may be small relative to the real space, people can easily miss small but significant changes. That is, for large workspaces it may not be possible to provide a useful representation in a miniaturized form. Although detailed views of the workspace may be made scrollable, either people can forget to scroll around or worse they may not even be aware of the existence of missing information (Plaisant, Milash, Rose, Widoff and Shneiderman 1996). One solution is to employ abstraction mechanisms or selective omissions (Card, MacKinlay and Shneiderman 1999) as a means to represent a large but still coherent amount of information in the small space of the overview. Abstraction mechanisms can combine (abstract) smaller related changes into a higher-level change. Examples of how changes can be combined as an abstraction will be discussed in Section 4.4.2 when I talk about semantic filters. Selective omissions can exclude certain information from the overview if it is judged irrelevant or less important than others. Further details about how certain information can be filtered will be covered in detail in Section 4.4.

An alternative approach to overviews uses focus and context (Furnas 1986) to display both overview and detail in a single display. Changes near the focus are displayed in the clearest and most detailed form. Less detailed change awareness information will be provided for changes further away from the central viewing area. The further away from the central viewing area, the less detailed will be the change awareness information provided. Many different research visualization methods have been developed around this idea. One example is the Perspective Wall, which uses 3D to

naturally visualize focus and context (Mackinlay, Robertson and Card 1991). The three panels illustrated in Figure 4.3 define a contiguous folded wall containing linear information stretching from left to right. The front center panel shows details. The side panels show similar information but its visibility depends upon how far back on the wall it is located. Thus, the front panel (the focus) shows details while the side panels show context. By making the area of interest (in this example the front panel) stand out more it makes information easier to acquire because it brings to the forefront information that the person is focusing on while keeping less relevant information on the side. The person does not have to do as much mental shifting as with zooming. Providing the surrounding context of a location where a change occurred makes it easier to understand the circumstances behind the change, thus making the information easier to interpret.

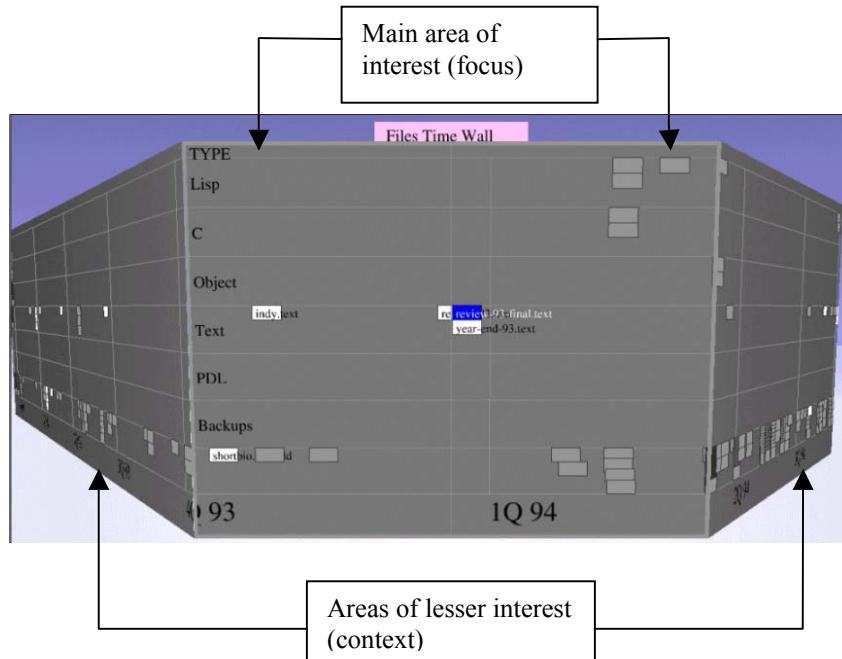


Figure 4.3: The Perspective wall as an example focus and context representation (Mackinlay, Robertson and Card 1991).

4.2.2 Representing information from the three workspace perspectives to reduce acquisition and interpretation costs

In Section 3.3, I described three different workspace perspectives: person-based, artifact-based and workspace-based. Requiring a person to query for changes in a way that

doesn't match his or her perspective makes it harder for that person to access change information. For example, an individual with a person-based perspective will find it awkward to have to check each artifact just to determine what another individual has changed. The cost of acquiring the necessary information is increased because they have to make several queries for just one piece of information. Also, it will be more difficult to interpret and relate information about what a specific individual has changed when the display of that information is scattered among the different artifacts that were changed by that person.

From the artifact-based view, a person should be able to query any object in the workspace and determine the details about its changes. We saw an example of this in Figure 4.2b when the user mouses-over an object. With the person-based perspective, one should be able to explicitly query for those changes made by another individual. One strategy is to provide some way of explicitly representing the other collaborators as entities in the workspace. Figure 4.4a shows mock-ups of how images can be used to represent other people, and how these images can be queried.

We see that when an image for a person (in this case 'Jim') is selected, the objects changed by that person (and only that person) are colored red. In this example, people's representations are literal, separate and permanent. The literal representation (the images) permits easy recognition of known collaborators. They are located separately and permanently to provide a ready list to the viewer that doesn't clutter the actual work area.

The actual changes are symbolic, situated, and passing, making it easy to query and interpret the information on-demand. Of course, other display approaches could move this into a different octant of Figure 4.1, e.g., clicking on an image could result in a literal, situated and passing animation that has the person's avatar (Benford, Bowers, Fahlen, Greenhalgh and Snowdon 1995) acting out all of the changes that he or she made. The avatar could be, for example a cursor containing the person's image that follows all actions in place.

The workspace-based perspective should permit one to determine changes made to the whole workspace or selected portions of it. For example, if one selects ‘all’ in Figure 4.4a, the view at the bottom will color all changed items in a room regardless of who changed them. Similarly, at a higher-level view of the workspace, when viewing multiple rooms, clicking on a particular room will show all the changes made only within that room (Figure 4.4b). Alternatively clicking on ‘all’ in this higher level view will show the changes of all the people made in all the rooms.

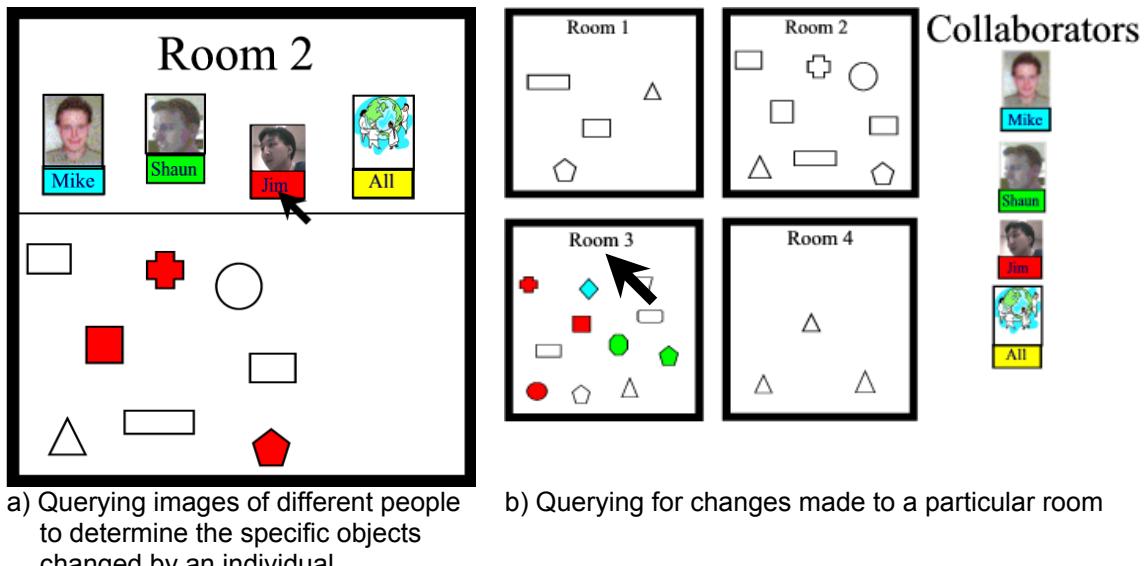


Figure 4.4: Querying for changes from two different workspace perspectives (mockups).

4.2.3 Reducing acquisition costs by displaying changes outside the workspace

Both of the above principles focus mainly on representation of change information as a person enters the workspace to probe for change information. However there are times when a person should know about incremental changes as others make them, or at key time periods. For example, an important bit of status information is ‘has anything of importance changed since I last looked’. Yet, if a person does not have the application open, they have no way of knowing this. Opening the application solely to see its change status requires too much time and effort for what should be a simple task.

One way of showing change status is to employ a *pull-based* approach. For example, *VSS*, Visual SourceSafe, (Microsoft 1992) allows a single person or group of people to track changes to one or more documents. (Figure 4.5). A person must explicitly request information about changes by invoking VSS. As different versions of a document are created, change information is logged into a file or visual list that can be reviewed by an interested party (the large background window in Figure 4.5). The person can also request specific details about a particular version on demand (the smaller foreground window in the figure). If many changes are taking place in the workspace then the log will tend to grow large requiring tedious searches in order to discover relevant information. Furthermore, viewing the log requires that the person invoke the VSS browser making it somewhat more bothersome to acquire the information than approaches described below.

An alternate approach to showing change status outside of the workspace involves *push-based* notifications, where a person is automatically informed of changes rather than having to explicitly request it e.g., MILOS (Dellen 1999). Typically, this is done through email notification, where a mail message is automatically sent as changes happen e.g., CVS (Berliner 1990). With these systems, a notification is sent when a significant event occurs, such as a new version being checked into the repository. The difficulty is determining when an event is significant enough to warrant the sending of a notification. If there is no effective way of determining significance, many change messages can accumulate over time demanding excessive reading and cleanup. Another problem is that the notification's symbolic and separate display may make individual events difficult to interpret and relate to the workspace context where the event occurred.

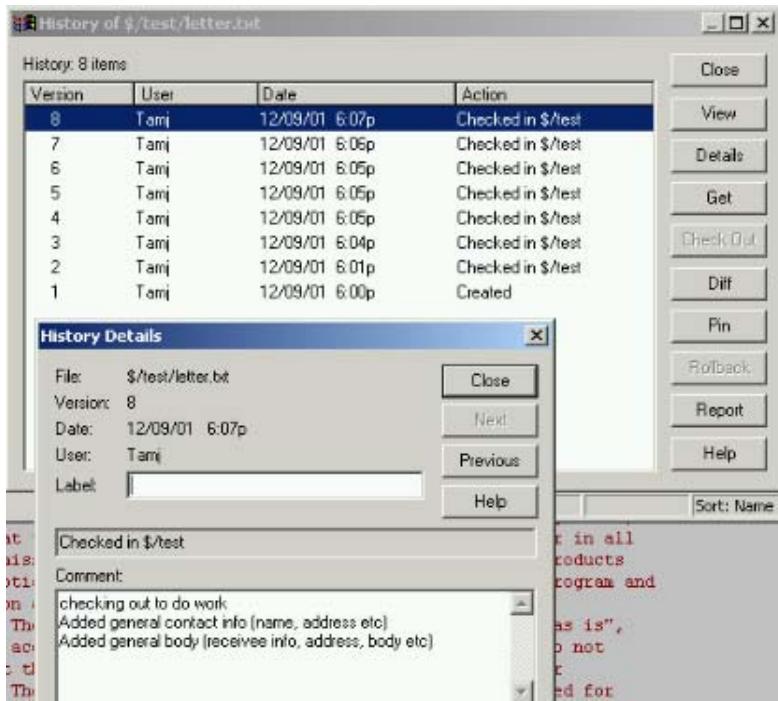


Figure 4.5: Change history of a document in a Visual SourceSafe repository (Microsoft 1992).

An example of a more lightweight push-based system can employ a permanent facility for displaying a high level overview of all of the workspace. For example if the collection is relatively small then thumbnail images or even real-time miniature radar views (Gutwin 1997) may be used to help people track changes made while they are away. Figure 4.6 provides an example. It depicts a small, always on, radar overview of a workspace divided into four rooms, where changed artifacts are colored with particular colors representing changes by different people. This display literally shows the workspace in real time making it easy to interpret what is happening. Although it is located separately from the workspace (since it runs outside of the application), it is still easy to figure out how the changes displayed in the radar view relate to changes in the workspace. Because its continuously running on a person's desktop (permanent persistence) it doesn't require a great deal of additional effort for the person to gain a sense of change activity. Subtle changes, however, may be missed in small overviews so an addition visual cue of size is employed. Notice in the figure how Room 2 and Room 3 are smaller than Room 4 and how Room 1 is the smallest of them all. In this, scheme

rooms grow larger as more changes occur in it, making changes increasingly more noticeable. This is a variation of the focus and context approach described by (Apperley and Spence 1982; Spence and Apperley 1982; Furnas 1986). The disadvantage to having a permanent desktop display is for workspaces that only periodically change, the user may be reluctant to devote screen real estate to it.

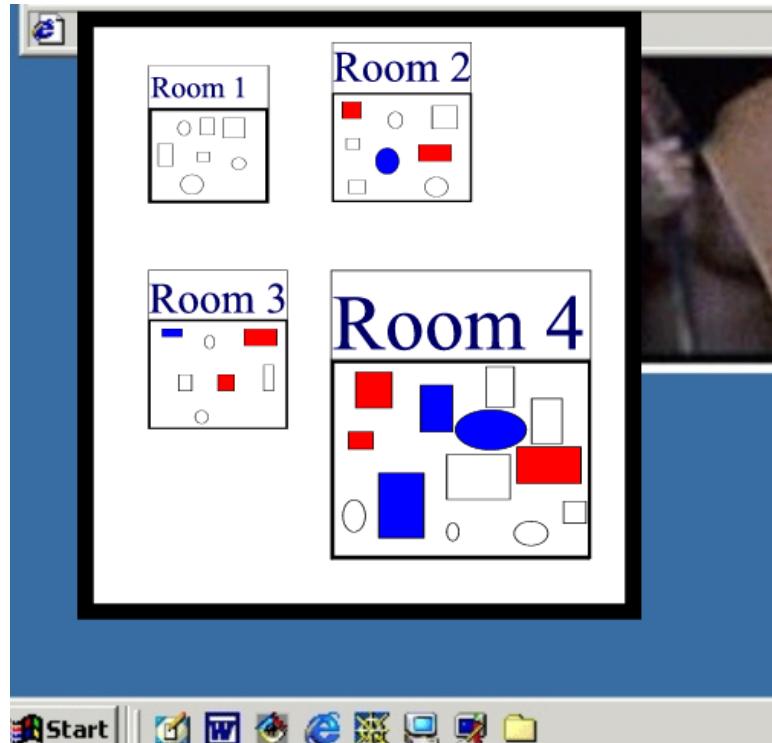


Figure 4.6: Display mechanism for showing changes made to the workspace outside of the application (mockup).

4.3 Displaying change awareness information

In the first section of this chapter I talked about the dimensions of displaying information about change awareness. In the second section I described ways of reducing the acquisition and interpretation costs of the information. In this section I will apply an approach used for representing cartographical information (Bertin 1967) in the representation of change awareness information. In addition, I will introduce two other mechanisms that can be employed in the display of changes: Kurlander and Feiner's (1991; 1993) *storyboarding technique* and *text*.

4.3.1 Bertin's visual variables and the display of change awareness information

Bertin (1967) defines a mark as something in space that is visible and can be used in cartography to show relationships within sets of data. He names the different ways that a mark can be varied as *visual variables*. Carpendale (2001) discusses and extends Bertin's original set of visual variables in terms of their use in information visualization. Here is Bertin's original set:

- *Position*, which are changes in the x, y, z coordinates of a mark (Table 4.1, second row).
- *Size*, which not only includes changes in height, width or area but also the number of times that a mark is repeated (Table 4.1, third row).
- *Shape*, which are changes in the form of a mark for a given size (Table 4.1, fourth row).
- *Value*, which are changes from light to dark (Table 4.1, fifth row).
- *Color*, which are changes in hue for a given value (Table 4.1, sixth row).
- *Orientation*, which are changes in angle (Table 4.1, seventh row).
- *Texture*, which are changes in fineness or coarseness of different patterns (Table 4.1, eighth row).

These visual variables have also been classified according to whether changes in a given variable enable the performance of different types of *visual interpretation tasks*, which are different characteristics of the visual variables (Bertin 1967; Carpendale 2001). The tasks of interest in this discussion are: *selective*, *associative*, *ordered* and *quantitative* tasks.

- Changes in a visual variable are a selective task if changes in the visual variable alone allow an object to be immediately stand out from a group (Table 4.1, second column).
- Changes in a visual variable are associative if a change in the visual variable alone allows a collection of objects to be grouped together (Table 4.1, third column).
- Changes in a visual variable are quantitative when changes in the visual variable alone allows for a numerical reading (Table 4.1, fourth column).
- Changes in a visual variable support an ordering task when changes in the visual variable alone allow a group of objects to be ranked (Table 4.1, fifth column).

Bertin also described a fifth type of representation characteristic, *length*, which describes the amount of variation that could be distinctly perceived by changes in the visual variable. I have chosen not to include length in my discussion because in terms of the 2D structured drawing programs that I have been discussing in this thesis, the amount of change variation that people may be interested in tracking is fairly small e.g., adding new objects, modifying or deleting existing objects. Consequently, the ability of a visual variable to handle a length type of visual interpretation task is less important than the other tasks and will not be discussed here.

Changes in visual variables can be used to best support the particular visual interpretation task needed for change awareness in a given circumstance. When a person desires to have important changes in the workspace stand out, he or she may engage in selective visual interpretation tasks. Tracking change awareness information will require associative visual interpretation skills when a collection of related workspace changes are linked together (i.e., to be viewed as one related ‘chunk’). The quantitative type of visual interpretation tasks indicates the number of changes that have occurred in the workspace. Finally the ordered task can be used to rank workspace changes for example, by their time of occurrence or by importance.

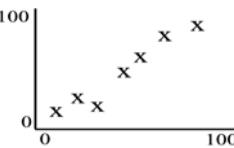
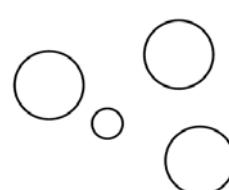
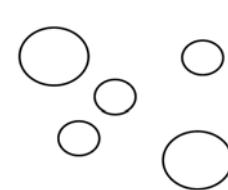
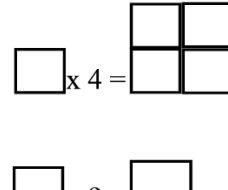
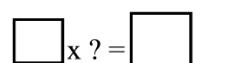
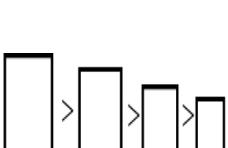
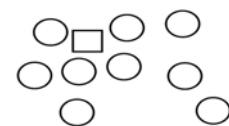
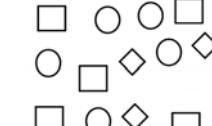
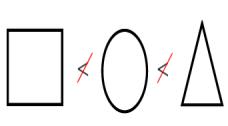
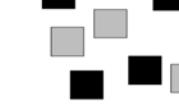
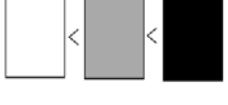
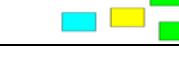
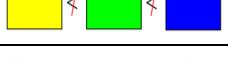
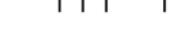
	Types of visual interpretation tasks.			
	Selective	Associative	Quantitative	Ordered
Position				
Size			 	
Shape				
Value				
Color (hue)				
Orientation				
Texture				

Table 4.1: Summarizing the effectiveness of different visual variables for different visual interpretation tasks, adapted from Bertin (1967) and Carpendale (2001).

Table 4.1 shows that not all the visual variables are effective for all the different types of visual interpretation tasks. In the discussion below, I will be referring to the cells of this table while describing how each visual variable may be used to represent change awareness information. Care should be taken when using visual variables in

existing applications to represent workspace changes because sometimes these visual variables can be manipulated by users and/or already have some underlying meaning (as discussed below).

Of all the visual variables, position is the most versatile and can be used for all four visual interpretation tasks (2nd row). Changes in position allow us to select the individual rectangles in the upper left and lower right corners (2nd row, 2nd column) or to differentiate the two groups of rectangles (2nd row, 3rd column). Position can be utilized in order to communicate specific numerical values or to rank information (2nd row, 3rd and 4th columns). Position can also be an effective means of communicating change awareness information. The simplistic approach for representing changes to a diagram involves either isolating or grouping changed objects using position in order to make them stand out or become related together. The danger is that spatial layout is a property that is typically under the control of the user so that rearranging the objects to communicate changes may alter the meaning of the diagram.

Position may be used to effectively communicate changes when spatial position is not a visual property under the control of the user, such as the scenario depicted in Figure 4.6, which shows radar overviews of a project. The position of these overviews is under the control of the application rather than the user so that position can be used as an alternative to size in the representation of changes. For example, all the radar views could start out at the bottom of the window in a line (Figure 4.7a). As changes occur within a room, the radar view of a room would shift towards the top of the window (Figure 4.7b) to provide the viewer with a quick idea of how rooms ranked with respect to the number of changes made within each.

Size is effective for representing change awareness information for selective, associative and ordinal visual interpretation tasks (3rd row). However, size is less effective for communicating quantity. While it is easy to notice that the four instances of the rectangle represents four times as many values, determining quantity for subtle differences in area is more difficult (3rd row, 4th column). We saw earlier in the chapter

with Figure 4.6 how size can be used to represent information about the quantity of changes made to a workspace. Much like position, because size is a property that is often under the control of users of 2D graphical applications, care must be taken when representing changes in this fashion.

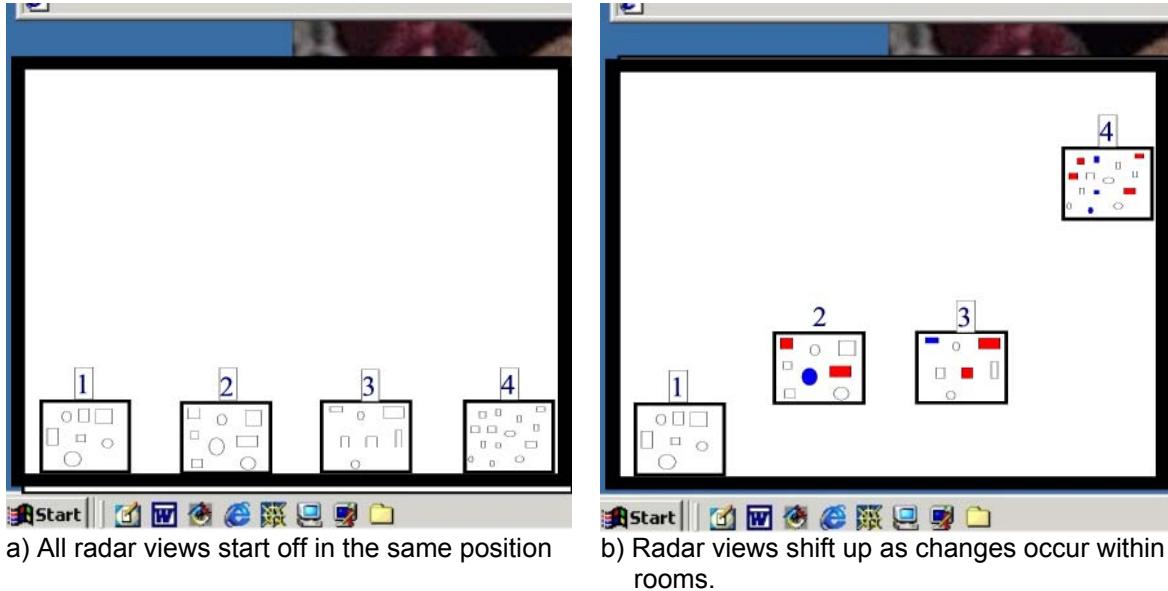


Figure 4.7: Using position to display the number of changes (mockup).

When there are few objects in a diagram, shape is an effective way of making individual objects stand out or for grouping categories of objects. However, it becomes increasingly difficult to associate objects together or to select individual ones as their number increases (4th row, 2nd and 3rd columns). Shape cannot be used to communicate quantity or order (4th row, 4th and 5th columns). As with the other visual variables, it may sometimes be difficult to use shape to communicate change awareness information because again shape may already have an inherent meaning within a diagram (Figure 4.8). In the example shown in Figure 4.8a objects change in shape according to the person who made the change. If objects already have a predetermined shape (Figure 4.8b) then altering the shape of these objects may appear somewhat jarring. It may be possible to ‘tag’ changed objects with different shapes to convey change awareness information as was done with the Model Integrator function in Rational Rose (Figure 2.12c). However, this requires that the shapes are made smaller (so as not to have the

shapes that convey the change awareness information overshadowing the original objects), but one must be cautious because if the change indicator (the shapes) are too small then the information less obvious and requires more effort to extract.

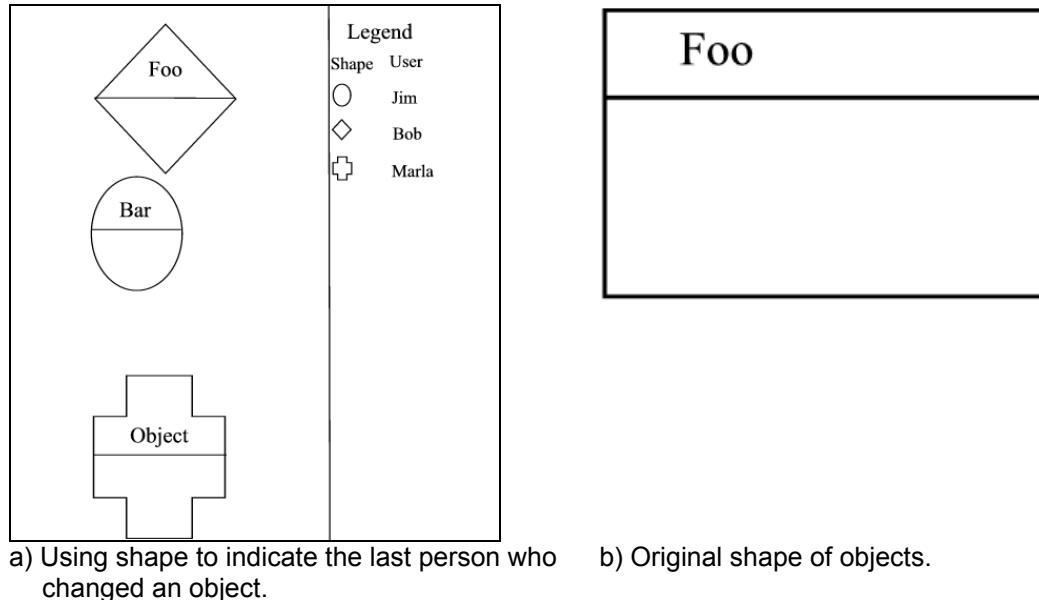


Figure 4.8: Using shape to represent changes.

Changes in value can make individual workspace changes stand out, or associate related workspace changes into groups (5th row, 2nd and 3rd columns), or can indicate some sense of order for workspace changes (e.g., from dark to light in the 5th column). For example, we could map value to roughly indicate how nodes in a concept map editor ranked according to the number of changes made to a particular node (Figure 4.9). The greater the number of changes, the darker would be the value of the node so that we can immediately see that node ‘Cats’ had changed more than node ‘Dogs’ and that node ‘Animals’ has undergone the most changes of them all. Node ‘Pets’ and ‘Whales’ have not changed which is indicated by their brightness.

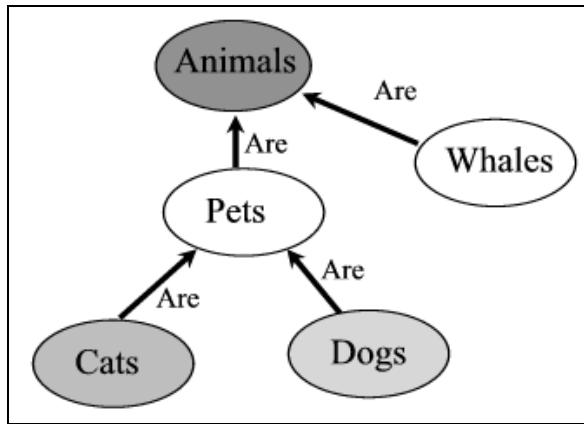


Figure 4.9: Using value to communicate how nodes ranked according to the number of changes made to them.

We tested the use of different levels of color saturation and value in a study (described in Chapter 5). The greater the number of changes made to an object, the lower would be the level of the saturation and value (and the darker object would look). Participants could easily distinguish the relative number of changes to classes from the color level – its situated, permanent display made this easy to do. However because no legend was provided, many people mistakenly thought that the different levels of saturation and value represented different types of changes vs. the number of changes, likely because the level was symbolic vs. literal, and we had not explained its meaning. Again, it may not be possible to use value if it is a user controllable property or if it already has inherent meaning in an application, which is quite typically the case in many 2D drawing applications.

Color is somewhat less versatile than value in that it can only be used to associate related workspace changes or select particular changes in the workspace (6th row) and is not ordered. However, color has been successfully employed in an existing change awareness tool, Seesoft (Eick, Steffen and Sumner 1992), which was described in Section 2.2.3.

Like color, orientation can be an effective way of making individual objects stand out or for grouping categories of objects (7th row, 2nd and 3rd columns). Figure 4.10

shows an example how changes made to nodes in a concept map editor can be all grouped together.

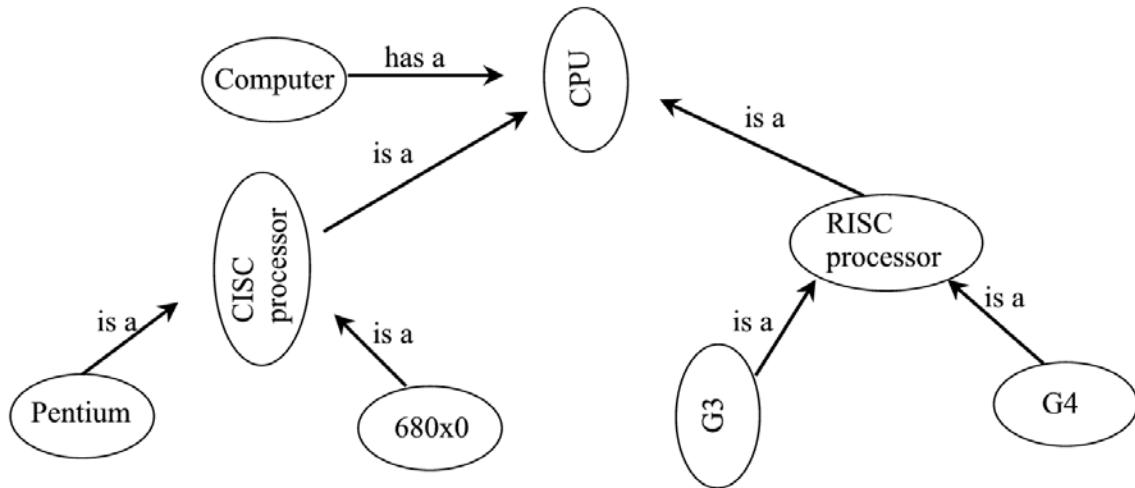


Figure 4.10: Using orientation to represent change awareness information.

It less likely that orientation, unlike other visual variables such as color, will be a user-controlled visual property. However, orientation may be a less effective mechanism for communicating changes for directed graphs such as concept maps and UML diagrams because the arrows already provide a sense of direction that may conflict with the orientation used to communicate workspace changes (Figure 4.11). Although it can be seen that nodes ‘James’ and ‘School’ are at a different angle than the other nodes, the orientation of these two nodes may conflict with the sense of direction conveyed by the arrow between the nodes. Also, it is obvious from the diagram that certain angles may make it hard to read text.

Like shape, color and orientation, texture can be used to associate groups of workspace changes together or to select out individual workspace changes. Figure 4.12 shows an example where a collection of related documents are shown together in miniaturized form. Unchanged documents are represented by a blank rectangle. Applying a different texture to the rectangle indicates changed documents. We can immediately see in the figure that Documents 1, 2, 6 and 12 are the ones that have changed.

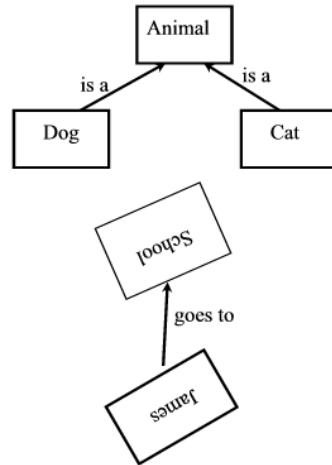


Figure 4.11: An example where orientation to communicate change awareness information may not work (directed graphs).

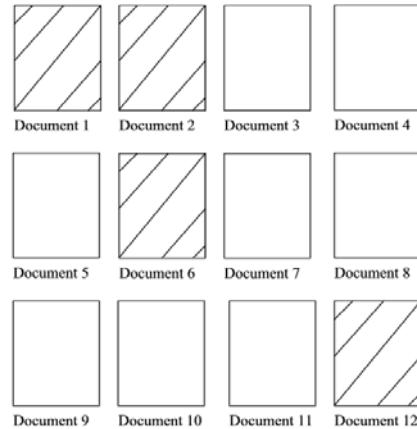


Figure 4.12: Employing texture to display change awareness information.

Although texture is less likely to be a visual property that can be manipulated by users, it must be used carefully. Inappropriate use may result in: vibratory effects (Bertin 1967) that makes viewing changes painful (Figure 4.13a); or textures that appear out of place in an application (Figure 4.13b). While it may be possible to tag objects with different textures to communicate workspace changes, much like the case of tagging objects with shapes, it may make the information about changes harder to notice.

To Bertin's original set of visual variables, Carpendale (2001) added motion. Movement can be used for either selective or associative visual interpretation tasks. In

addition, Gutwin (1997) suggested that by having workspace artifacts “act out” in real-time the changes made to them (in the form of animations), workspace events are more noticeable. For example, rather than having a deleted node in a concept map editor suddenly disappear, he proposed animating the deletion process so that the node first grows in size (to draw attention to it) and then have it gradually shrink out of existence (node 2 in Figure 4.14). Similarly, in our study (described in Chapter 5), we discovered that animated replays are an extremely clear way of making individual changes stand out. However, purely literal animations can include unimportant actions e.g., as a person tries out several possibilities before determining a final course of action. Animations could be ‘smoothed out’ where unimportant events are not shown. As will be seen in Section 6.3 this can have a large effect on the utility of a change awareness tool employing animations.

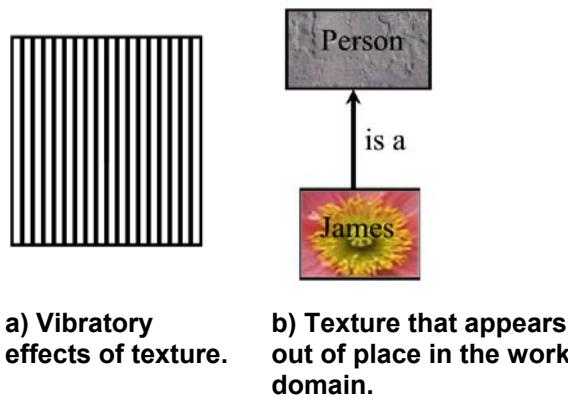


Figure 4.13: Misuse of texture to represent changes.

In addition to employing visual variables to represent change awareness information, I propose two other mechanisms: storyboarding, and text. Both have been successfully employed in other contexts and may prove promising in the display of changes. They will be described in the following subsection.

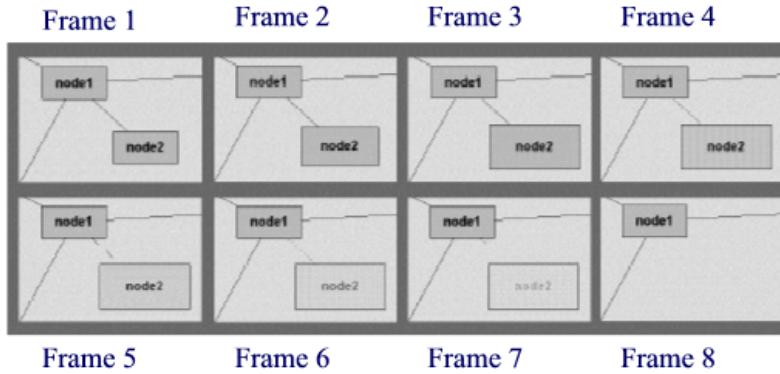


Figure 4.14: Animation of a node being deleted in a concept map editor (Gutwin 1997).

4.3.2 Other display mechanisms: storyboarding and text

Chapter 2 described Kurlander and Feiner's (1991; 1993) storyboarding technique to portray the interaction history of an application. For change awareness, a series of panels can capture how multiple people change the workspace over time; each panel captures a significant change in the workspace. The person sees the incremental sequence of changes in this series and can examine in detail the before and after view between panels. Storyboarding thus provides a literal and permanent presentation because each panel shows a capture of the workspace. This quick overview was precisely one of the reasons why some of our test participants liked the use of storyboarding to represent changes in our study (Chapter 5).

The key problem in storyboarding is determining when an event is significant enough to warrant the creation of a new panel. If a new panel is created for every low level event that occurs in the workspace e.g., every time that someone draws a line, the many panels will make the storyboard tedious to view. The solution employed by Kurlander and Feiner (1991; 1993) combines panels by collecting similar actions e.g., all line creations that are part of the higher-level task of drawing an object are collected into a single frame. Panels can also exclude (or filter) changes e.g., when a person draws a line then immediately deletes it. However when panel combination and filtering is taken to excess, transitions from one frame to another may be too abrupt and a person may lose the flow between successive frames (as was the case in our study described in Chapter 5).

Also, subtle changes could be difficult to spot. For example, many popular children's puzzles ask viewers to find subtle difference between two nearly identical panels of a comic strip. The two example pictures in Figure 4.15, taken from a newspaper's weekend comic supplement are identical except for six differences. This search task is not always easy (try it!¹). By extension, detecting small changes between storyboard panels can also be difficult.² One solution is to use the other techniques described in this chapter to highlight differences between one panel and the next one in order to make each difference stand out.



Figure 4.15: Using before and after comparisons to spot differences between panels (courtesy of King Features Syndicate, Inc).

The second change representation mechanism that I propose is written text. Text is a complex and relatively complete symbolic notation. Its advantage is that it can be very descriptive and detailed. It is suited for describing abstract information such as temporal or logical data (Larkin and Simon 1987). However, the interpretation cost can be quite high because processing this type of information is, "...detailed, serial, low capacity, slow, able to be inhibited, conscious" (Card, MacKinlay and Schneiderman 1999: pp. 25). Text can either consist of short labels that symbolize an event (Figure 4.16a) or it can be

¹ The six differences are: girl's eyelashes and bow, dog's ear, man's eyes, hair and collar.

a lengthy, involved, and detailed account of events (Figure 4.16b). Text can be either situated e.g., by spatially locating change annotations on or near the object as in Figure 4.16 or it can be separate by displaying change documentation in a separate window. While situating change information is obviously desirable, the workspace can become cluttered unless a passing level of persistence is used. However, separate text means we have to somehow indicate what the text is referring to.

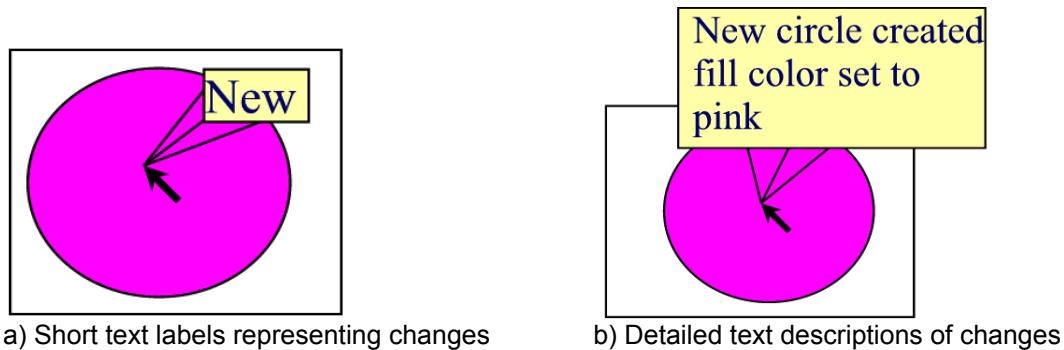


Figure 4.16: Different detail levels in the display of changes using text.

4.4 Filtering the display

In addition to the techniques described in Section 4.2, information filtering is yet another way of reducing the acquisition and interpretation costs of change awareness information. There are two different approaches to filtering information. *Boolean filters* display information if the filter criterion has been met and hide it otherwise. In contrast, *progressive filters*, allow for the progressive display of information, with the way that change information is displayed reflecting how closely that the filtering condition is met. After first introducing Boolean and progressive filters, I will discuss three filtering strategies: basic change filters, semantic filters, and hierarchical filters.

² During a quick test with three people using the above cartoon, it took almost a minute for them to spot three to four of the six differences in the panels shown in Figure 4.11. Only one person out of three participants spotted all six differences.

4.4.1 Types of filters

Boolean filters reveal information that strictly matches some filter criterion. If it does not pass the filter test, the information is not displayed. With progressive filters, change information is displayed gradually and reflects how closely that the filtering condition is met.

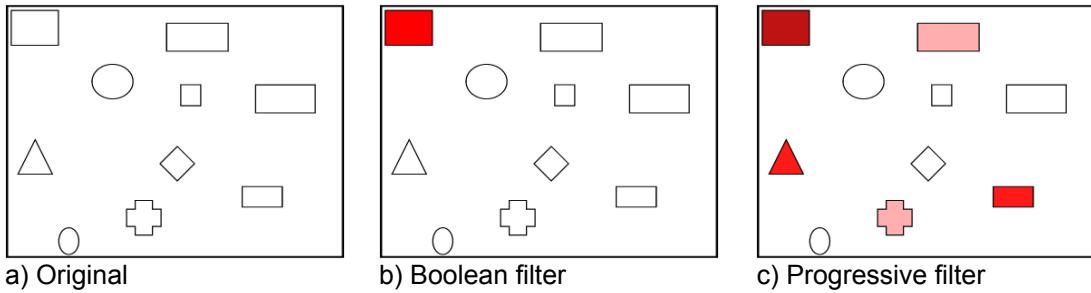


Figure 4.17: Different approaches to filtering

Figure 4.17 provides examples of both approaches. Figure 4.17a shows the final state of a drawing. Figure 4.17b applies a simple Boolean filter to this drawing, where the person is only interested in objects that were changed one day ago. The figure shows the one object that meet this criterion colored red. Figure 4.17c shows the progressive filter, on the same query where different levels of color saturation and value indicate the length of time that passed since changes were made. As before, objects changed within the last day are colored in deep red. Objects colored in lighter levels of saturation and value were changed more than a day ago, where the levels correspond to time. Other progressive values can also be represented e.g., saturation and value could be used to highlight change by author ranking, where authors that are more ‘important’ could have their presumably more important changes colored more deeply.

Ideally, both Boolean and progressive filters should be adjustable on the fly so that the person can dynamically make queries about changes, while immediately seeing its result on the display (Shneiderman 1984). This allows for the quick easy exploration of change information.

4.4.2 Filtering strategies

Three filtering strategies may be employed: basic change filters, semantic filters, and hierarchical filters. Throughout this section, I will use a running example to illustrate the difference between these three types of filters, where the example is an E-banking software project modeled using UML class diagrams (Figure 4.18). This project consists of four packages: UserGUIs, Security, Persistency, and Web (Figure 4.18a). Each package consists of a number of classes; Figure 4.18b shows the classes that comprise the UserGUIs package. Finally, each class consists of data and methods with the Logout class illustrated in Figure 4.18c.

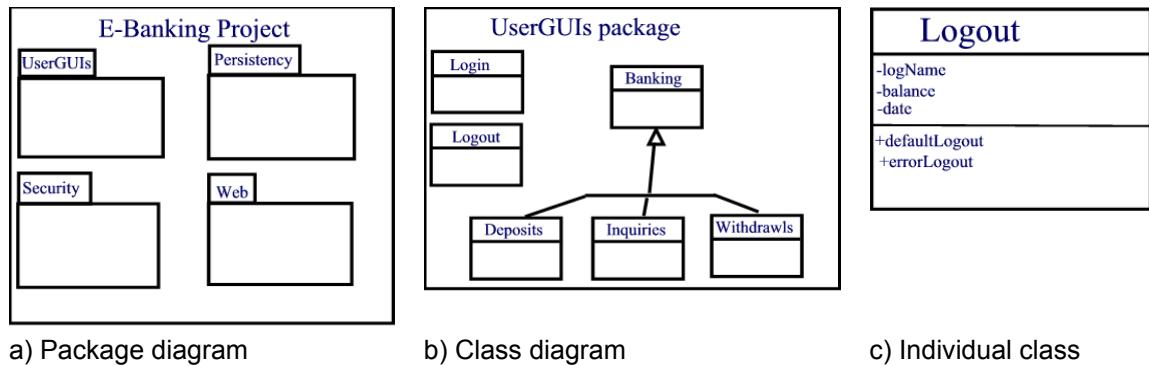


Figure 4.18: An E-banking software system modeled with UML diagrams.

Basic change filters. One of the first ways in which change information can be filtered is by the basic low-level and generic information associated with every change. This information relates back to some of the fundamental questions that a person can ask when trying to determine what has changed, which was already discussed in Chapter 3. Aside from linking them to particular categories of change awareness questions, they are generic as they are not specific to any task domain and can be applied to a wide variety of work scenarios. For example, Figure 4.19 shows the classes in package UserGUIs. Classes that have been changed are colored in red. In Figure 4.19a, color is applied to all classes that were changed e.g., by movement, by changed relations between classes and so on. Figure 4.19b gets more specific and only applies the color indicator to classes that have undergone change within one day. While the basic change filter does allow one to

filter out some irrelevant information we could do better if knew something about the domain semantics and its structure.

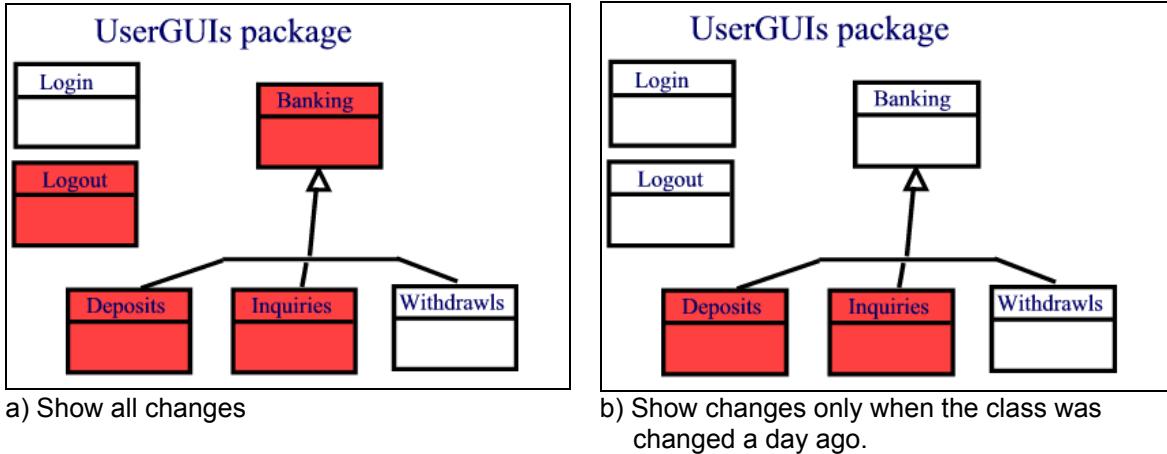


Figure 4.19: Using basic filters to screen changes.

Semantic filters. Within every task domain, there exist a set of *semantics*. The semantics of domain determines the meaning behind the actions that people take and we can filter by these semantics instead of low-level primitives, as is the case with the basic change filters.

For example, Figure 4.20 supplies a before and after version of the classes in package “Persistency”. This drawing could have been made with a simple 2D graphical editor using basic objects (rectangles, an arrow, and text). Alternatively, it could have been made with a UML editor where objects are domain specific e.g., classes and a generalization (parent-child) relation. With the generic 2D editor, changes may be described as a spatial movement of an arrow and a rectangle (containing the text ‘Oracle’) moving from one part of the diagram to the other. In a UML editor, changes are described in terms specific to the domain e.g., Class ‘Access 2000’ was changed from being a child class of ‘Access97’ to a child of ‘DataBase’. This is a semantic change, which better describes what is going on than the spatial move.

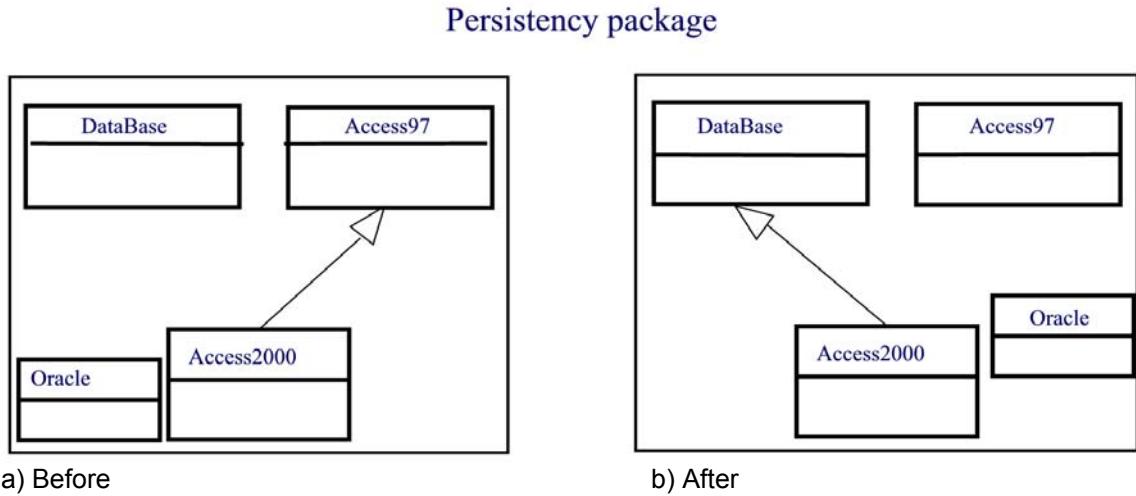
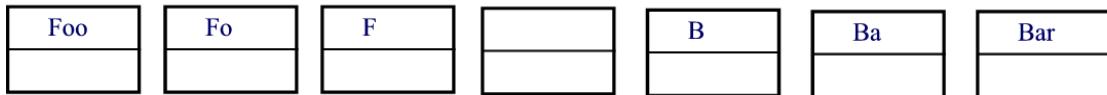


Figure 4.20: Before (a) and after (b) versions.

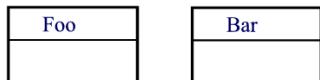
A semantic filter also removes irrelevant changes e.g., syntactic changes that have no semantic effect in the task domain. In Figure 4.20, if software engineers filtered classes to show only those whose meaning had changed, then the spatial movement for class ‘Oracle’ that happened between (a) and (b) would not be shown.

Semantic filters can also combine several lower level primitive actions together into one higher-level semantic action. This filters out information about atomic actions that are of minor semantic relevance to people in a task domain, instead embedding them into a single higher-level action that gives its smaller atomic actions meaning. For example, Figure 4.21a employs the storyboarding technique to represent atomic changes made to the title of a class in a UML class diagram where a new panel is created for every changed character. Obviously, all these low level character changes are all part of one higher level change i.e., renaming the title of the class from ‘Foo’ to ‘Bar’. Figure 4.21b combines all these changes where a single before and after comparison shows changes to the class diagram. Presenting a good abstraction of the data i.e., symbolic form can help simplify and organize complex information (Card, Robertson and Mackinlay 1991; Resnikoff 1989). McCaffrey (1998) also noticed that people tended to organize and classify changes into different categories. Of course, determining what changes can be

combined will be highly dependent upon the semantics of the task domain; if we over-combine, meaningful information could be lost.



a) Representing individual changes with a storyboard panel.



b) Combining changes.

Figure 4.21: Filtering changes that are depicted with storyboards.

The major advantage of semantic filters is that it allows queries via the semantics of a domain e.g. ‘show me all changes made to a class vs. show me what has changed’.

Figure 4.22 again shows the classes of package UserGUIs. This time the semantics of the domain are used as a mechanism for filtering changes. Color is applied only to the classes which had a relation changed so that some of the changes portrayed in Figure 4.19 will not be shown in Figure 4.22. For example, while class ‘Deposits’ is colored red in Figure 4.19 indicating that it was changed, there is no color indicator in Figure 4.22 because none of its relations were changed.

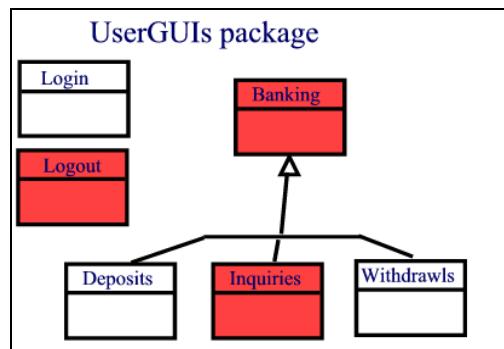


Figure 4.22: Semantic filtering of change information.

Hierarchical filters are a special form of semantic filter. For hierarchically organized domains, the structure of the hierarchy can provide the means for the filtering of change information. The idea is that changes are presented in a gradual fashion. A

rough picture of changes is provided in an overview at the uppermost levels of the hierarchy, and details are seen as one descends into the hierarchy.

One way in which this can be done is to have information about smaller related, changes combined and represented as one larger higher-level change. This combination of information, known as the formation of *macro-operators* (Nilsson 1980; Korf 1987) is how experts have been found to solve problems that they have experience with and why they solve these problems so quickly (Anderson 1983; Larkin, McDermott, Simon and Simon 1980). Consequently if the right information elements are combined it may actually speed up the assimilation of information about changes as well as helping to avoid overload.

For example, using an abstracted bird's-eye overview of a hierarchy, a person can gain an overview of changes and probe deeper into the hierarchy for further details. The deeper that someone goes into the hierarchy, the more change information he or she will be provided with. While similar in idea to the context plus detail approach mentioned in Section 4.2.1, the difference is that the display will abstract the information into semantic hierarchical chunks.

Figure 4.23 illustrates the hierarchical filtering of changes. Figure 4.23a indicate through red coloring that the UserGUIs package has changed somehow but no details are provided. The user selects that package, and now sees all the classes within 'UserGUIs' where only changed the classes that were changed are filled in red (Figure 4.23b). Based upon this information the person selects the class 'Logout' and now sees the changed fields of the class displayed in red and in a larger font (Figure 4.23c). When the person mouses-over a changed field, additional change information appears in the form of a pop-up box.

The benefit of the hierarchically filtered view of changes is that changes are presented abstractly and gradually allowing people to go from an overview and selectively query changes in depth, making it easier to interpret what is happening. As

indicated earlier, many of our test participants (Chapter 5) wanted changes presented in a summarized, abstract fashion.

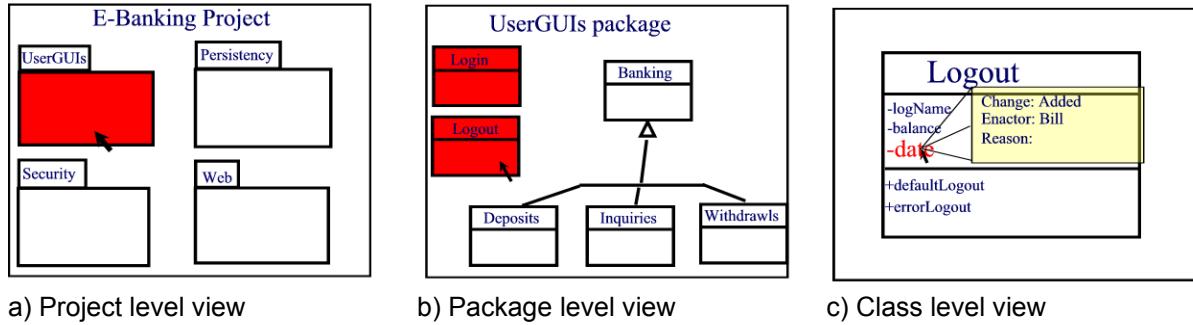


Figure 4.23: Hierarchy of UML package and class diagrams.

4.5 Summarizing the important concepts in the representation of change awareness information

Four things should be considered when representing awareness information about workspace changes. First, there are the different ways in which changes may be displayed (display dimensions), which include the presentation (symbolic or literal), placement (situated or separate) and persistence (permanent or passing) of changes. The placement and presentation dimensions were adapted from Gutwin's (1997) framework for workspace awareness, while I added the persistence dimension to support specific aspects of change awareness.

The second consideration is to make the information readily available and easily understandable by reducing its acquisition and interpretation costs. If a person has to engage in a time-consuming process while using an application just to track changes, he or she simply may not bother. The person may bypass the change awareness support provided by an application and instead resort to manual techniques such as visual comparisons or by just asking the person who made the changes. One way of reducing the acquisition cost of change information is to display this information outside of the workspace. In addition, visualizations, such as overviews and details (Card, MacKinlay and Shneiderman 1999), or focus and context displays (Furnas 1986) can be used to

reduce acquisition and interpretation costs. The three workspace perspectives (artifact, person and workspace based) described in Chapter 3 must also be accounted for in the acquisition and display of change information.

The third consideration is that the display mechanism must represent the informational elements in a meaningful fashion. In this chapter, I described how some of Bertin's (1967) original set of visual variables (position, size, shape, value, color, orientation and texture) and Carpendale's (2001) elaboration of this set and her additional visual variable (motion), can be used for different types of visual interpretation tasks that necessary in the tracking of change awareness information. In addition, other display mechanisms (storyboarding and text) also show promise as ways of portraying changes.

Fourth, in projects where many changes are made, it may be necessary to filter the information. Boolean filters display or block information based on whether they meet a filtering criterion. Progressive filters allow for the gradual display of change information depending upon how closely the filtering condition is met. Different filtering strategies include: basic, semantic, and hierarchical change filters. Basic change filters allow for rudimentary queries of changes, e.g. 'Has anything changed?' Semantic filters are specific to a task domain and allow for more sophisticated queries to be formulated, e.g. 'Have any class methods changed?' Semantic filters can also screen out irrelevant changes or combine several lower-level changes into higher-level changes. Hierarchical filters exploit the existing containment structure of the workspace to present changes in-depth in a gradual fashion making the information easier to interpret. A person starts by seeing an overview of the project and the changes made. Based upon this information the person can probe for further details by exploring deeper into the hierarchy.

Chapter 5

Investigating change display mechanisms

I briefly digress in this chapter, where I describe the results of an early formative pilot study in change awareness. In this study, I examined the effectiveness of several mechanisms for displaying change information, including: object animations, storyboarding, text labels¹, and the use of written documentation to describe changes. Each of these mechanisms represent an example of the placement and presentation quadrants for displaying workspace awareness information described in Section 4.1. However it does not include persistence; the need for this dimension only became apparent to me after many study participants complained about how changes shown within one display mechanism would not stay onscreen. This study was conducted before the creation of my conceptual framework for change awareness (Chapter 3) and the accompanying display concepts (Chapter 4). Consequently the results influenced the formation of the framework (e.g., the need to classifying change information) and the display concepts (e.g., the need for the persistence dimension). The complete study description is published as Tam, McCaffrey, Maurer, and Greenberg 2000.

5.1 Methodology

In this pilot study, we interviewed participants as they explored and used our simulated prototypes. This method is fairly standard practice in Human-Computer Interaction (Nielsen 1993; McGrath 1994).

¹ The original study incorrectly referred to this display mechanism as icons. This error has been corrected in this chapter although the published version still uses the original terminology.

5.1.1 Test participants

Nine test participants were recruited from several sources: a graduate program in Software and Computer Engineering, undergraduate Computer Science students, and programmers from industry. What is important is that all participants had UML experience.

5.1.2 Test materials

To ensure consistency between tests we laid out a specific set of test procedures (Appendix A1), which were carefully followed for each participant. In addition each participant was required to sign a consent form prior to the beginning of each test (Appendix A2).

Four versions of a UML editor were employed - a prototype mock up built via html image maps. All versions displayed UML class diagrams augmented with change information. Each version employed a different change display mechanism, which will be described in detail in Section 5.2.

In this study we developed three sets of questionnaires: *a pre-test questionnaire, a post-scenario questionnaire and a post-test questionnaire*.

The **pre-test questionnaire** was used to determine the participant's computer and UML experience (Appendix A3).

For the four mockups a generic **post-scenario questionnaire** (Appendix A4) was created and reused for each prototype. This questionnaire was used to determine participant's understanding of the display mechanism(s) employed, where we asked him or her to explain how they thought it worked. We asked specific questions about changes made to the UML diagram, including: what objects changed, how did those objects change, who made changes and when did these changes occur. The participants then rated the effectiveness of the display mechanism on a scale of 1 to 5, with 1 labeled

'useless' and five labeled 'very effective'. They also were to indicate why particular ratings were given. Finally we asked each person how he or she would augment the mechanism so that it would be more effective at portraying changes.

The **post-test questionnaire** was administered at the end of the test. This final questionnaire asked participants to compare the mechanisms employed by each mockup and to indicate which one they preferred. In addition it asked them to describe any other techniques (other than the ones represented with the four prototypes) that they would use to track changes in a group project.

5.1.3 Test procedure

For each participant we followed a fixed procedure (Appendix A1). We would begin by describing the purpose of the study to the participant. Then the participant would be asked to sign the consent form (Appendix A2). Next we determined the computer and UML experience of participants with the pre-test questionnaire (Appendix A3). After this the specific details of the test procedure was described to the participant. Each person was then shown a sequence of four task scenarios, each using a different change display mechanism (making this a within-subjects test). For each scenario, the participant interacted with a class diagram displayed within one of four versions of a mocked up UML editor.

Participants were asked to imagine that they were software designers working on a team project that they had left for a period of time. They had now returned and needed to determine the changes that had been made to UML diagrams by other members of the team. These changes were typical of the ones that a designer would make to a class diagram, such as adding new classes, or adding and removing the fields of a class.

For each scenario, each participant began by briefly exploring UML diagrams and the changes made to them as displayed by the particular version of the UML editor. We then interviewed the participant with questions from the post-scenario questionnaire

(Appendix A4). This procedure was repeated for all four display mechanisms. Finally the post-test questionnaire (Appendix A5) was administered.

All the scenarios modeled class diagrams in the same basic UML editor. To minimize learning effects between scenarios, however, diagrams differed by the specific set of classes used as well as the changes made to them. Finally we randomized the order in which the change display mechanisms were presented to participants.

5.2 The change display mechanisms

All versions of the UML editor used color to indicate change within a hierarchy of information. For example, Figure 5.1 shows a set of UML classes, where each class represents a hierarchy of elements (methods and data which are hidden in this view). The class “SystemGUI” is colored in yellow because it has not changed. The other three classes are colored with different saturation and color values of red, where the level of red indicates not only that a class has changed, but how much. In this case, color is used as a symbolic and situated display mechanism. Color was also applied to the relationships between classes to indicate when the relation had changed. Because all displays use color we expect that it provides an at-a-glance method of spotting changes over all classes. However, when a person needs to explore these changes, then he or she could probe for further details by using the particular display mechanism provided by the editor version, as described below.



Figure 5.1: Color and variations in saturation and value are used to indicate if a class has changed and by how much.

The four main change display mechanisms employed in this study were based upon the placement and presentation dimensions described in Section 4.1, and include: object animations, storyboards, text labels, and text documentation (Table 5.1).

		Placement	
		Situated	Separate
Presentation	Literal	Object animations	Storyboards
	Symbolic	Text labels Color	Text documentation

Table 5.1: Classification of the change display mechanisms employed.

Object animations are a literal and situated change display mechanism that allowed participants to view changes made to a class diagram, one class at a time (Figure 5.2). Clicking on a class would start the animations that replayed all the changes made to it. For example, in Figure 5.2 we see the editor replay the sequence of changes that were made to class Client. We showed animations on an object-by-object basis because McCaffrey (1998) noticed that test participants who wanted to see information about a single change to an object would become annoyed when they had to view an animation of changes to all objects.

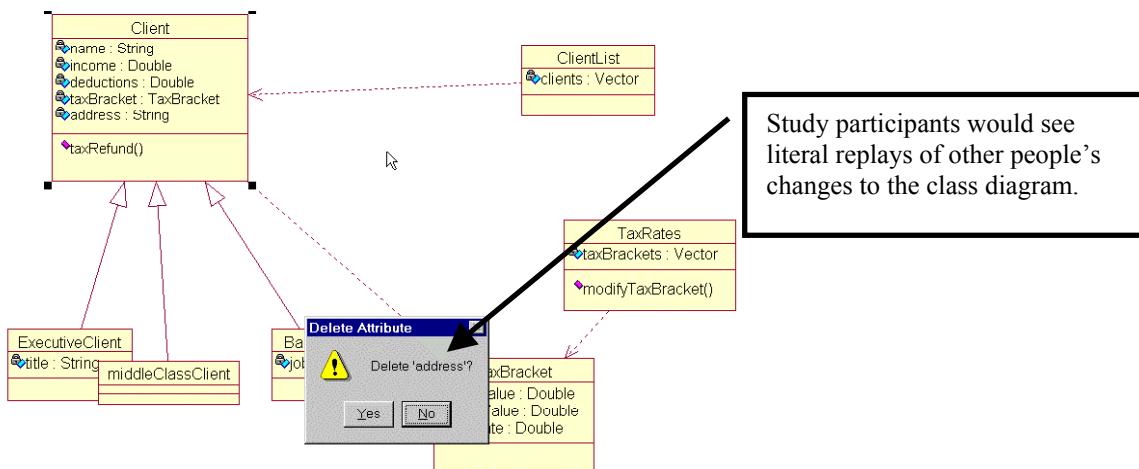


Figure 5.2: Animated replay of changes (literal, situated)

Storyboards created a panel for every single change event that was made anywhere (top of Figure 5.3). Each panel presents a small image of the entire diagram after the change. The images are a literal presentation, while the detached location of changes

classifies the placement as separate. Participants can click on a panel to see details of the change in the main view (bottom of Figure 5.3). The title bar of the detailed view described who made the change and when e.g., the panel shows that a change was made by Lorin McCaffrey on March 3, 2000.

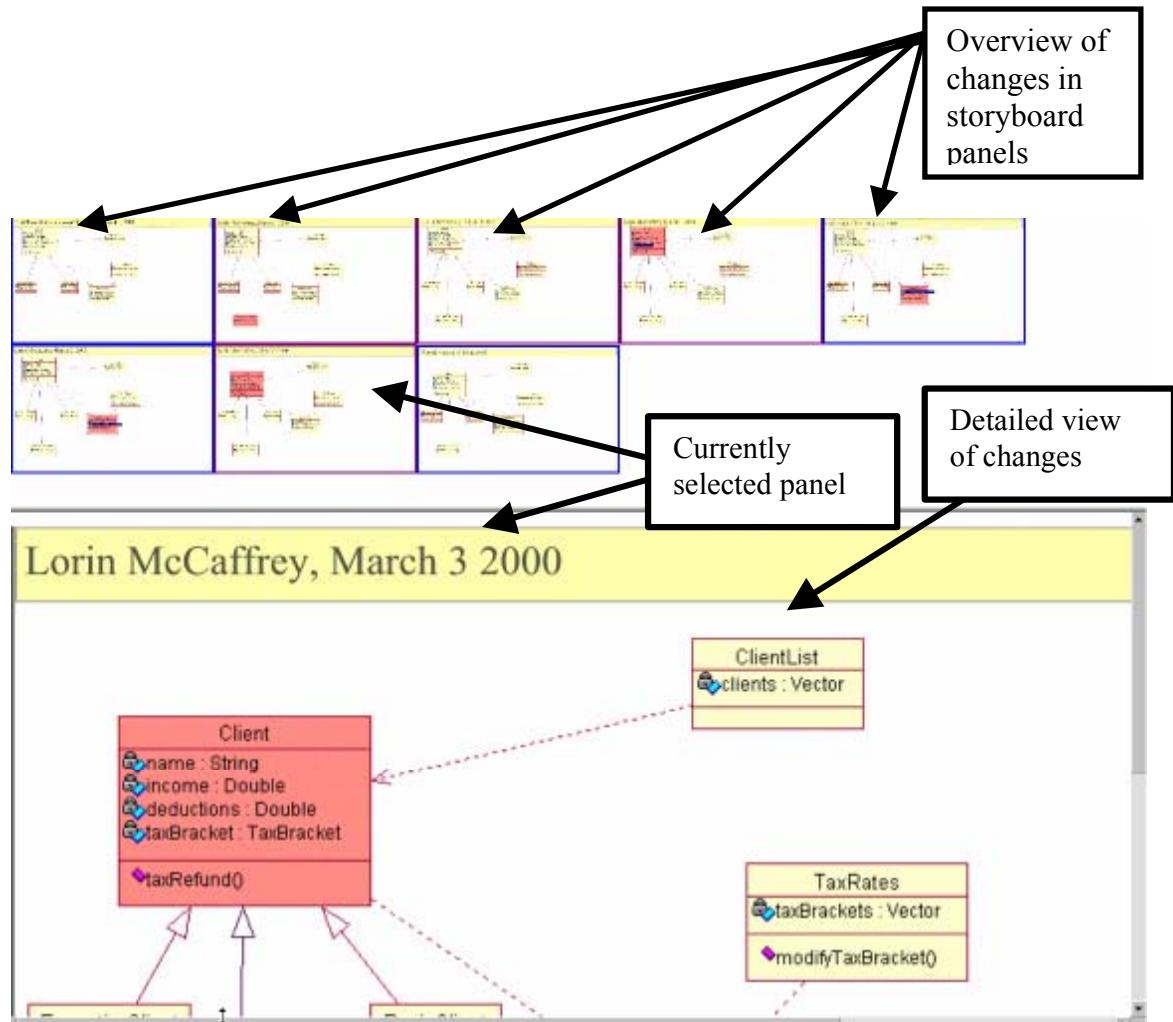


Figure 5.3: Overview and detailed view of changes using storyboarding.

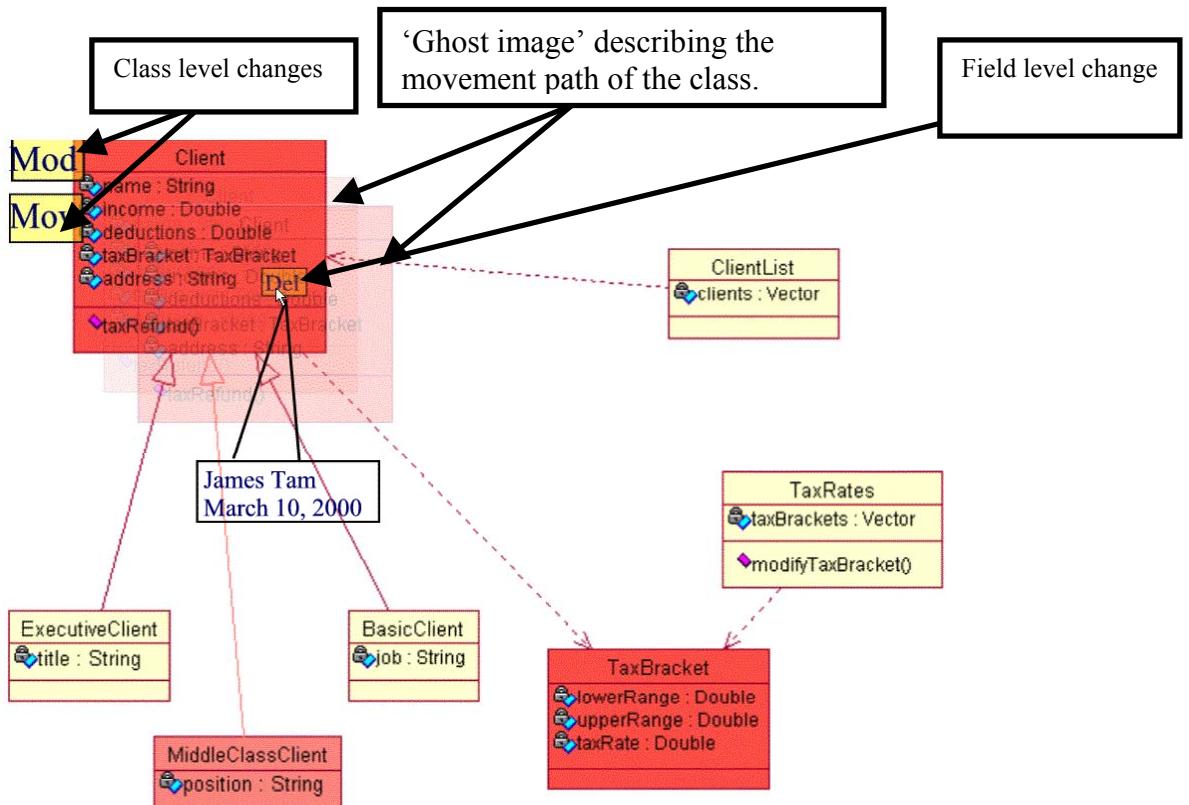


Figure 5.4: Text labels used to represent changes.

The **text labels** consist of brief (three character) descriptions of changes made at both the class and the method/data level: “Add” indicating an addition. “Del” indicating a deletion. “Mov” indicating that something had moved. “Mod” indicating that existing information was modified (such as a method or class name being renamed). These small labels would be attached to the fields of the class (data or method) that had changed. Figure 5.4 (class Client near the top left) shows that “address”, a field of class “Client”, has a “Del” label beside it indicating that this field was deleted from the class. Changes that were relevant to the entire class would then have a larger text label attached to the left of and slightly outside of the changed class box (“Mov” and “Mod” for class Client). Each text label could be clicked on, and some of them provided further change information about who made the change and when. For example, the person has clicked on the DEL labels, and he now sees that James Tam deleted the address field of the class Client. Because we could not explain the spatial movements of classes in a text popup, we supplemented the text labels by ‘ghosted’ images of the class showing its previous

locations. For example, we can see in Figure 5.4, the movement path taken by class Client is represented using these ghosted after-images. Text labels are a symbolic representation of changes that are situated with the class or field that was changed.

Finally, **text documentation** displays longer text descriptions to describe changes (Figure 5.5). When a person clicks on a changed class, written documentation explaining the change details would appear at the bottom of the screen. Details indicate what happened to the class, when it happened, and the name of the person who made the change. For example, in Figure 5.4 we see that the person has selected the class “TaxBracket”, and the documentation explains that this class had two of its fields renamed by James Tam in March 2000. Again the text is a symbolic mechanism but in this case it is located separately from the changed part of the diagram.

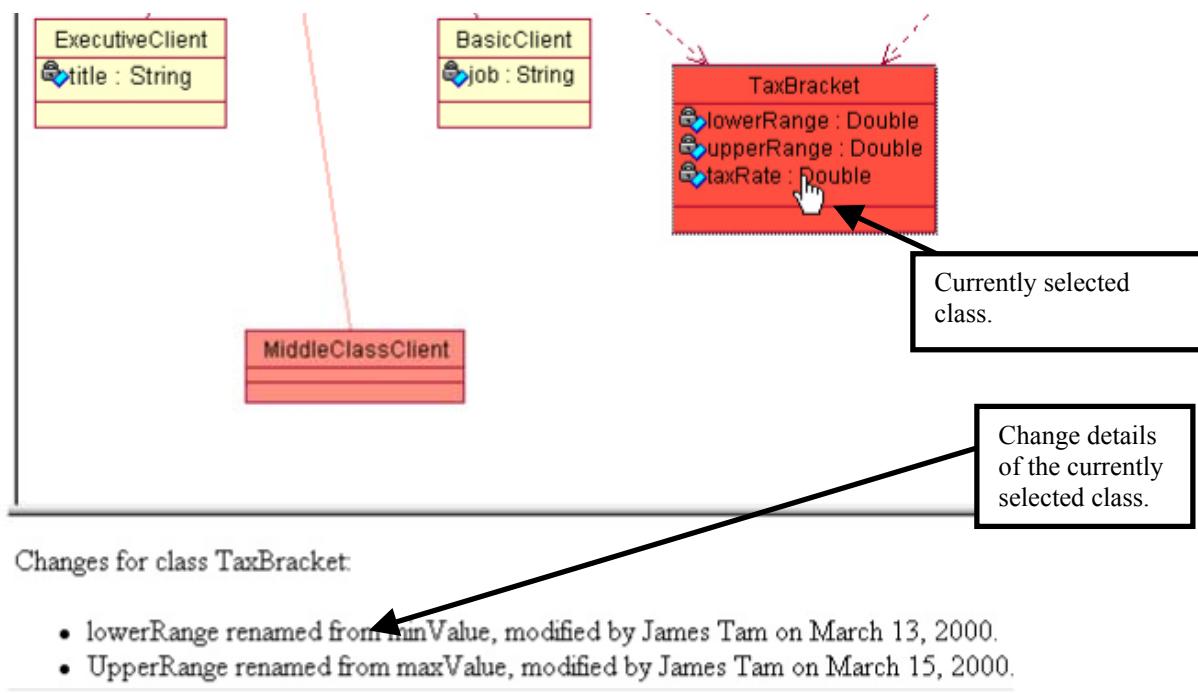


Figure 5.5: Using written documentation to represent information about changes.

5.3 Results and discussion

In the first subsection below, I discuss the subjective ratings and rankings that participants gave to the four display mechanisms. Also, I describe some of the reasons that participants gave for rating a particular mechanism the way that they did. With the next subsection, I discuss some of the insights that were gained about specific display quadrants for representing changes. Afterwards, I cover the study findings that were not unique to a specific mechanism or quadrant. I close by raising several confounding factors in this study.

5.3.1 Findings about the display mechanisms

Slightly over half of the test participants stated that they liked the documentation method best (Table 5.2). However the comparative ratings of the four mechanisms, depicted in Figure 5.6, suggest that this is not a strong preference. We see that the highest rated method, documentation, received a rating of just over 4. This was slightly below the highest possible rating of 5, which equated to ‘very effective’ on the questionnaire. Text labels, however, were not far behind, with a rating of approximately 3.6. Even the least favored storyboarding mechanism, received a rather neutral rating of 3, which is approximately a point less than the most popular method.

Display mechanism	No. who rated the mechanism as best
Animated replay	1
Storyboard	1
Text labels	2
Documentation	5
TOTAL	9

Table 5.2: Summary of the preferences of test participants.

Participants typically stated that why documentation was preferred was because it was “fast and efficient”. In particular, participants said that you could see all of the changes related to a class with only a single mouse click. Several participants also thought that it provided the richest potential for describing changes. It should be noted, however, that several participants had trouble understanding changes that were spatial in nature (such as movements) using this mechanism.

The main benefit participants ascribed to the text labels was that it too allowed them to quickly recognize how many changes occurred and what those changes were. Again many participants had trouble understanding changes involving movements using text labels, even with though additional graphical cues showed these movements through ghosted after images.

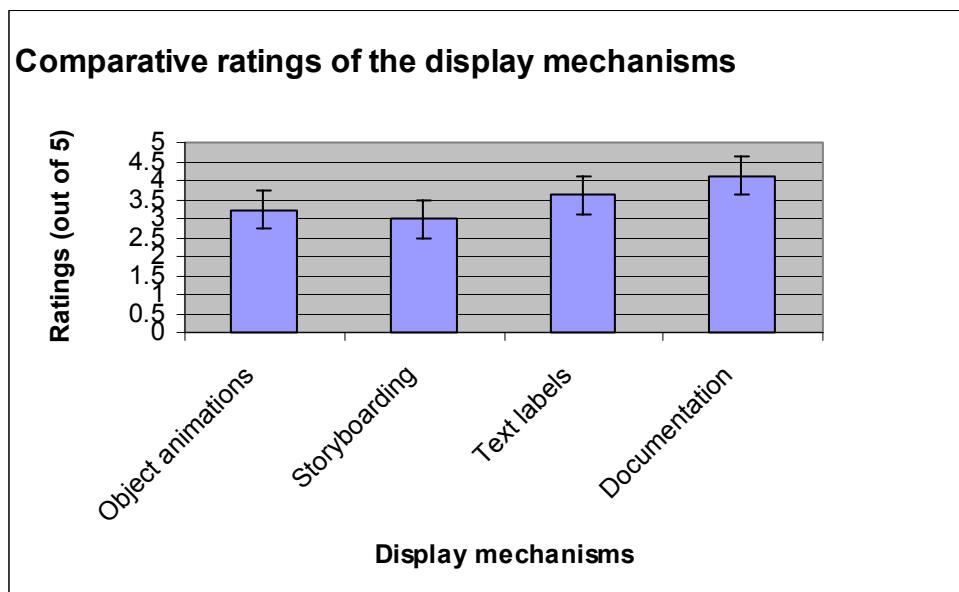


Figure 5.6: Comparative ratings of the different display mechanisms.

People said how they liked the chronological overview provided by storyboarding and how they could quickly determine the number of changes that occurred (by counting the number of panels). Participant’s major complaint against storyboards was the number of before and after comparisons that were required. People found that the miniaturized panels were too small to do before and after comparisons where the single detailed view

was the only view in which they could clearly see all the particulars of the changes. This complaint may have been compounded by the slowness of the implementation: a person was required to click on a miniature storyboard panel, and then wait for a few moments to load the image into the main panel. As with the previous techniques, people found it difficult to detect and understand spatial movements in our particular implementation of storyboards. This is likely caused by the coarse granularity of the storyboards; the only way that a person could determine that something had moved was to notice and visually compare the difference of position in the before and after panels this was made worse by the small size of the miniaturized panels.

Participants commented that most actions are immediately recognizable using animations. However since changes did not stay onscreen, participants had to constantly re-watch the animation to determine all the changes. This was worsened by the fact that the prototype did not allow participants any control over: playback speed, frame rate or level of detail, stopping, pausing, reviewing or fast-forwarding through animations. Participants also wanted to view the current version of the class diagram as the animation was replayed so that they could do some before and after comparisons.

5.3.2 Findings about the display dimensions

Other problems noted by participants about the various change techniques likely apply to the individual display quadrants as a whole, as listed below.

Situated vs. separate: Many participants complained that the gaze shifting required when using the separate displays (storyboards and documentation) to be a lot work. However, a few also mentioned that if many changes occurred the situated display would get quite cluttered.

Literal vs. Symbolic: Many participants expressed a preference for the symbolic presentations over the literal ones in terms of speed. It seems that the symbolic methods (text labels and documentation) allow viewers to quickly understand the abstracted representation of complex changes. However, there is a tradeoff. While the information

presented by the literal presentations sometimes took longer for participants to absorb, they were more accurate in their responses.

5.3.3 General findings

A common occurrence observed with many test participants, irrespective of display mechanism, was that they either did not like or were confused by changes involving the movement of classes within the diagram. Three participants mentioned that *hierarchical* rather than spatial moves should be tracked. A hierarchical move occurs when a portion of code has been deleted from one class and added to another class. A spatial move takes place when the class itself is spatially shifted within the diagram. While this makes sense within the hierarchical constructs of UML diagrams this is not applicable to other domains e.g., simple drawing editors.

Participants also wanted some control over prototypes, where they could configure the display mechanisms. In particular, there were many suggestions about adding filters that would allow them to selectively block irrelevant changes (such as movements).

Most participants thought displaying changes in a hierarchical form was a good idea. They liked how color provided a quick overview of changes, and how it let them pinpoint where they could probe for further details.

5.3.4 Limitations of the study

There were a number of simplifications and assumptions that were made for the purposes of this study. Since they may have a potential confounding effect on the results I will list them in the remainder of this section.

Although participants are told to imagine that they are software designers who are working on a real-world project, for this study these people did not have the opportunity to see the class diagrams before the changes were made. Thus they represent the ‘worse case’ of forgetfulness, as they could not rely on their memory of what they had done before to guide some of their explorations.

The prototypes also presented varying amounts of change information. For example, animated replays did not show information about who made changes, or when these changes occurred. Nor did the replays provide documentation explaining why changes were made. Storyboards omitted information on process history and showed only before after comparisons of changes. This may explain why many participants found it difficult to understand some types of changes, such as deletions and movements, with storyboards. Not all of this information could have been added to particular display mechanisms, for there was no clear way of representing some types of information within them e.g., we couldn't find a literal way displaying 'why' changes were made.

The prototypes did not always isolate different techniques, and several often appeared together. For example, text labels were supplemented by graphical cues; storyboards were annotated with text descriptions; all mechanisms were supplemented by color.

Finally some of the complaints that were expressed by participants about various display mechanisms were related to our specific implementation. For example, participants indicated that they liked documentation better than storyboards because they could determine changes more quickly. This may be due to the fact that the particular implementation of documentation used in this study would show all changes with a single mouse click, while the storyboards required multiple mouse clicks. That is, our implementation made it more effortful to probe for details using one mechanism when compared to another.

5.4 Conclusions

This pilot study explored the value of four mechanisms for representing changes in a software design project, where each method was chosen from a different quadrant of the placement/presentation grid. Each mechanism was supplemented by the use of color to provide an overall view of changes, and people could probe for details about changes by using the four change display mechanisms.

Although this was a formative pilot study conducted with only a limited number of participants, a number of common themes stood out. We found that symbolic displays had the potential to represent overviews of large amounts of information by abstracting many changes into a compact form. However, literal displays provide the benefit of clarity. Because the testing was conducted in limited simulations, some features were not included in the prototypes. For example, the animated replay was a passive viewing of changes rather than an active process of exploration. Because both mechanisms appeared to have some promise in the representation of change awareness information, I took things a step further by building an actual working prototype that employed these mechanisms (described in Section 6.1).

We found that situated displays require less gaze shifting and are easier to place in context. However, as mentioned by several participants, if many changes were to take place the danger of cluttering the workspace becomes apparent. Conversely, with the animated replays, there are times that needed information was no longer displayed onscreen. Thus, I added the persistence dimension (described in Section 4.1) to classify how long change information should stay onscreen.

There is also a need to filter changes. Even in the small example project that we tested, which consisted of a single class diagram, most participants expressed the desire to be able to screen out information that they thought was irrelevant (such as spatial movements). There should be an even greater need for filtering in larger projects consisting of multiple diagrams. As a result of this need I developed several types of filters and filtering strategies (discussed in Section 4.4) as well including different filters in a prototype change awareness tool (described in Section 6.1).

Finally, this study indicated the value of displaying changes in a hierarchical form. The prototypes used in this study allowed only a single level in the hierarchical presentation of changes: color was used to indicate changes at the class level, further change display mechanisms were used to communicate changes at the method or data level. These results, plus the research of others describing the value of using overviews

(Schneiderman 1996; Furnas 1986; Card, MacKinlay and Shneiderman 1999) suggest the need for the hierarchical presentation of changes not only on object basis but on a project level basis showing an overview of changes made to a collection of diagrams (as described in Section 4.2). In the more fully developed prototype (described in Section 6.1) I further explored the utility of employing hierarchical views of changes where I allowed for multiple levels in the viewing of changes.

Chapter 6

Appraising the framework and display concepts via a sample change awareness implementation

In the earlier part of my research, I spent the majority of my time developing ‘PastDraw’ a 2D structured drawing application augmented with change awareness. At that time, I had little understanding of exactly what this system should support in the way of change awareness information, and how I would display it. My approach was to add a smorgasbord of change techniques into it, where I concentrated mainly on system implementation and architectural details for supporting change information. After mostly completing the first version of this drawing editor, I realized that there was something not quite right about how I was portraying change awareness within it. Yet, I could not precisely articulate what was wrong. Consequently, I abandoned the implementation in order to develop the framework described in Chapter 3. I also outlined the important concepts necessary for displaying this information in a meaningful and useful fashion (Chapter 4).

In this chapter I return to the implementation, where I now critique ‘PastDraw’ in terms of what change information is tracked (through the framework), and how that information is represented (through the display concepts). I also show how the framework suggests how PastDraw could be modified to improve its support for change awareness.

The point of this critique of PastDraw is to demonstrate the value of the framework and the display concepts, where I show how the principles from the previous chapters can be used to appraise change awareness tools and to suggest how they can be improved or redesigned.

6.1 Introduction to the base system and a usage scenario for PastDraw

I begin by introducing the PastDraw system: how it was built, and a scenario of its use.

The Tigris research community (Tigris 2000) created the *GEF*, Graph Editing Framework, (Tigris 1995) an open source library of Java (Sun Microsystems Inc. 1996) classes that can be used by software designers to develop many different types of 2D graphical applications. These applications could range from basic structured drawing tools to more specialized tools such as the UML editor Argo/UML (Tigris 1999). The GEF library includes a demonstration drawing application (Figure 6.1), which allows end users to create and manipulate basic 2D shapes (circles, rectangles, text boxes, lines etc).

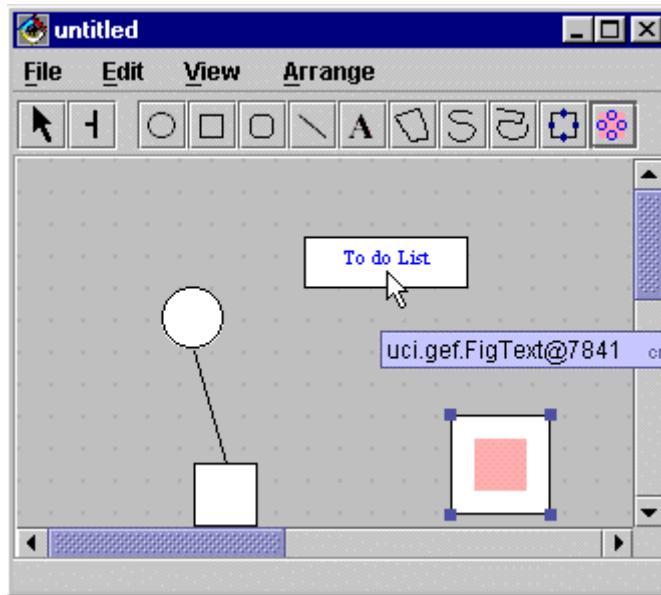


Figure 6.1: A basic drawing application created from the classes in the GEF (Tigris 1995).

Because the GEF provided both an open-source class library and a drawing application, I decided to extend it to support and illustrate change awareness, where it would capture and display the past changes of other people. As already mentioned, much effort went into the implementation and architectural details. I named the resulting system PastDraw. In the remainder of this section, I introduce the features of PastDraw through a typical usage scenario. This sets the stage for later discussion.

Jenn and James are drawing pictures of their dream house using PastDraw, where they have explored many different layouts of rooms and furniture. They have been working on the drawings for several weeks. When they both agreed on a change made to a diagram, they used the system's "accept changes" feature so that PastDraw would no longer track and display information about that event. The last time that they agreed on the changes that they each made was a week ago.

One morning James wakes up to find a note from his wife saying that she was up most of the night working in PastDraw because she had an inspiration for some new designs.

Being intensely curious about the changes made, James loads up the house drawing in PastDraw (Figure 6.2). He first looks at the overview of the house displayed in the project overview (top right corner), which shows thumbnail images of the two levels of the house. To the left of the project overview is the real-time radar view (Gutwin 1997) of the diagram that is currently loaded, the document overview. Within these two overviews, objects that have been changed are colored to indicate which PastDraw user made the changes. We see in the figure that objects changed by James are colored blue (or black if the image is in black and white) while the ones that Jenn changed are green (or medium gray). Together, the project overview and the document overview provides him with a rough idea of where changes occurred, what objects were changed, as well as who made the changes.

James then looks at the detailed main view, shown in the large area in the left part of Figure 6.2, in order to find out more information about changes made to the first floor. Since James is already aware of his own changes, he selects the filters (right middle) to filter changes by person so that PastDraw will show only Jenn's changes. Through the animation controls (bottom middle of figure), he sets the change display mechanism to animations and then pushes play to get more information. These animations literally replay what happened during the last week

James tries to find the spot where Jenn started working last night but he discovers that PastDraw will only replay the changes starting from a week ago. In this version, he cannot fast forward the slider to start displaying changes from last night onwards. Nor does there exist the ability to filter the changes made during the week to show only the ones from last night.

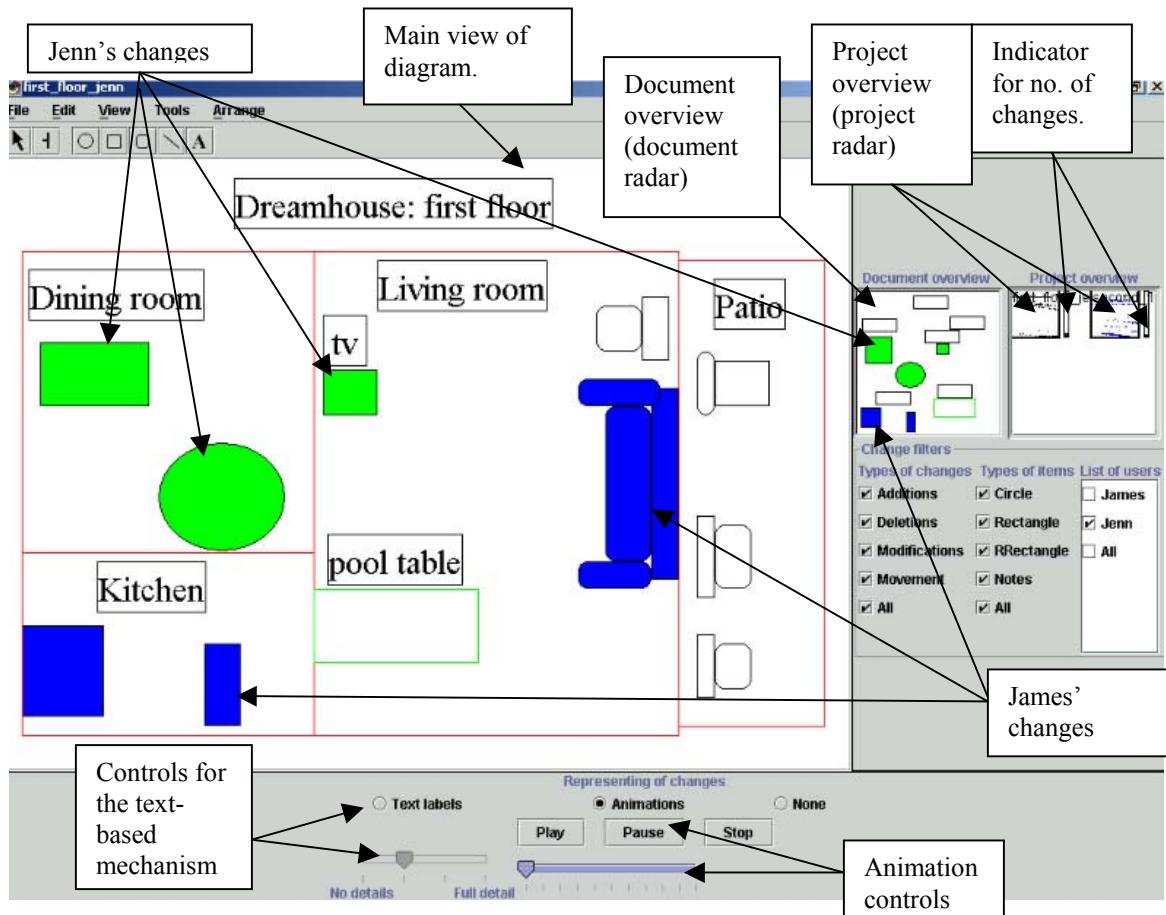


Figure 6.2: Overview of the main features of PastDraw.

In addition, Jenn had many false starts so much of the replayed sequence shows activities that do not really go anywhere. James finds these animations just too literal a recount of what happened: even small formatting adjustments (the movement and resizing of objects) are replayed. He sets the filters again, this time to screen out movements and modifications from the animation but now he finds that too much information is missing. He no longer gets a sense of the editing flow, and he still cannot

determine what are false starts, and what are fruitful actions. Since James does not know what objects Jenn has changed, he does not touch the controls to filter by object type.

James decides that he needs a better general understanding of the changes that Jenn made during the night. There are just too many changes being displayed by the animations, and what's worse, each change stays on-screen only briefly so that James cannot get a general sense of what happened. While the thumbnail overviews give some change information ('where', 'who' and the specific objects changed), James wants more details.

James switches the change display mechanism to show "text labels" that briefly describe each change that has occurred over the last week (Figure 6.3). We see, for example, by the "DEL" indicator that his pool table (center) has been deleted from the living room. The diagram also includes outlined/faded graphical cues that provide additional details about 'how' certain types of labeled objects changed. With the living room television, for example, an arrow shows the previous location of the television as well as its final destination. Also two outline images and some resizing arrows show the dimensions of the TV before and after it was resized. In contrast to animations that show all of the intermediate locations, the approach employed with the text labels and cues show only the initial and final view of the television's movement path.

James is somewhat overwhelmed by all this information, so he uses the slider control for the text labels (Figure 6.2, bottom left) to adjust the level of detail shown by the labels and images. We see in Figure 6.4 the effect that the different slider levels have on the display of changes for text labels in the detailed view. "No details" turns off the display of all change information (Figure 6.4a). The next level provides only color cues for changes (Figure 6.4b). The level after that adds text labels that describe changes (Figure 6.4c). "Full detail" augments the text labels with graphical cues like the arrows used for spatial moves.

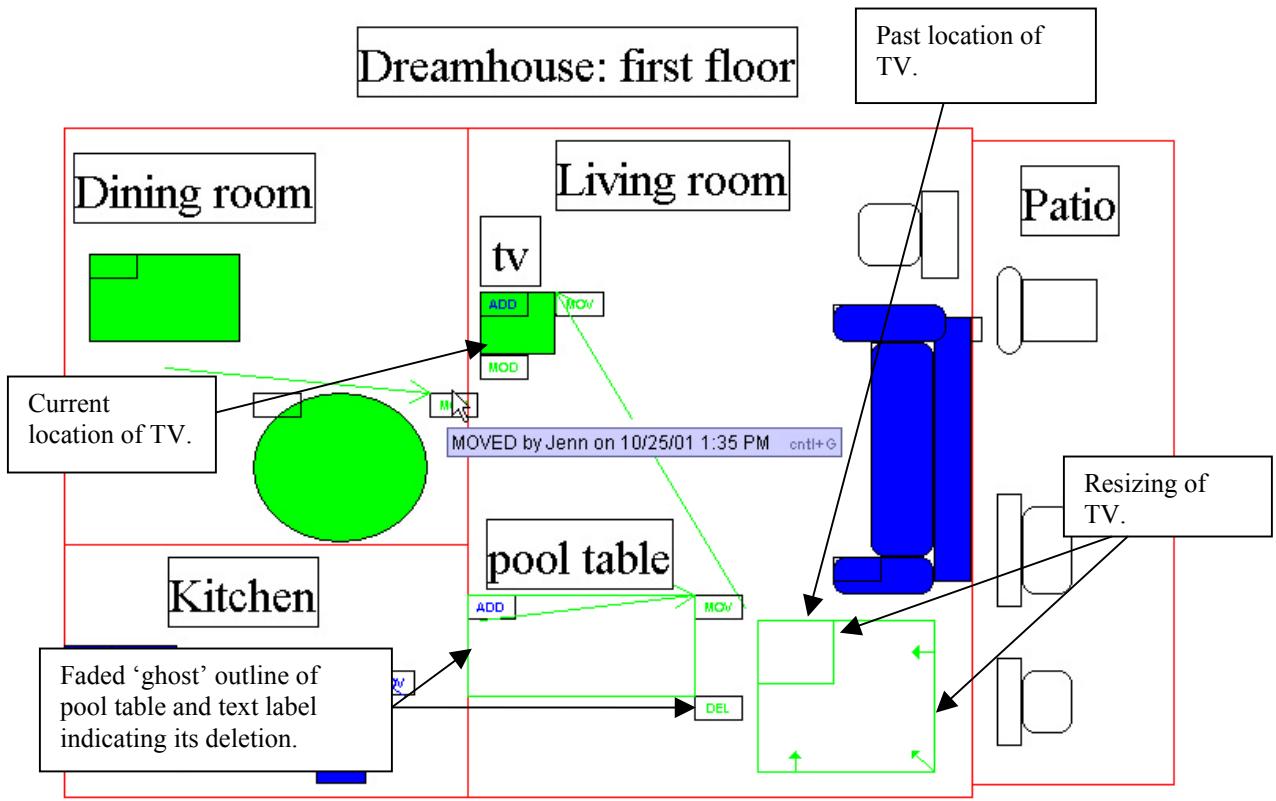


Figure 6.3: Text labels describing the changes.

After a few minutes of exploring this changed view of the first floor, James thinks he knows what has changed. He then looks at the thumbnail image for the second floor, and he notices a fair amount of change coloring. He clicks on the graphic to load it up in the main view (Figure 6.5). James notices that most changes occurred in the pool room (lower left), where a lot of new furniture has been added to it. He also notices that the pool table, which had been deleted from the first floor, is now added here. Out of curiosity why Jenn added the table, James mouses-over the label and sees that Jenn has added a note explaining what happened here i.e., that this room would now serve as a recreation room (bottom of Figure 6.5).

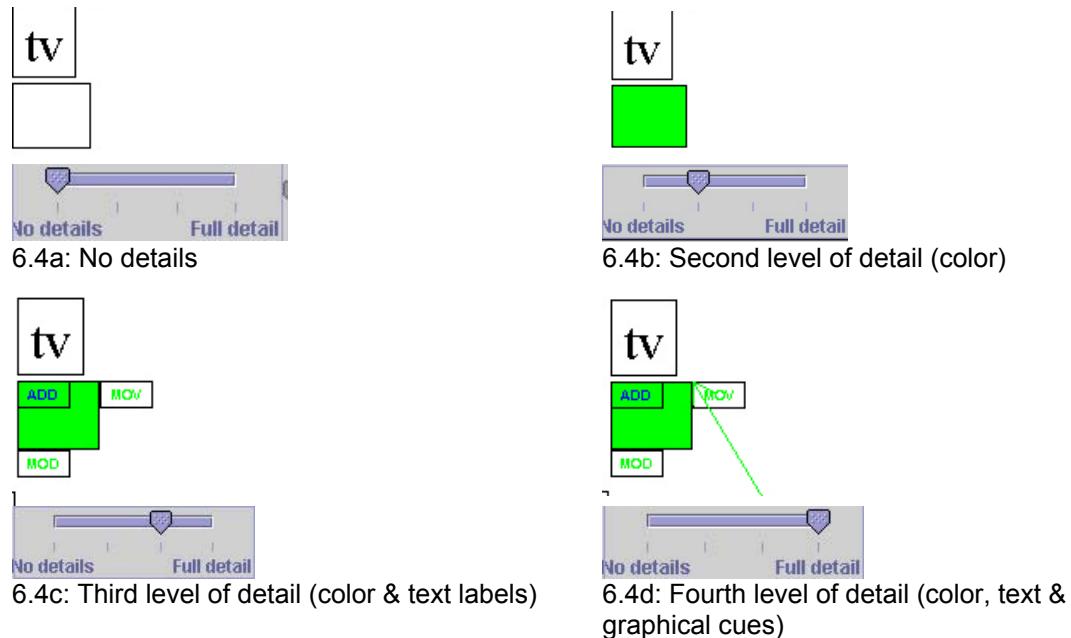


Figure 6.4: Multiple levels of detail provided with text labels.

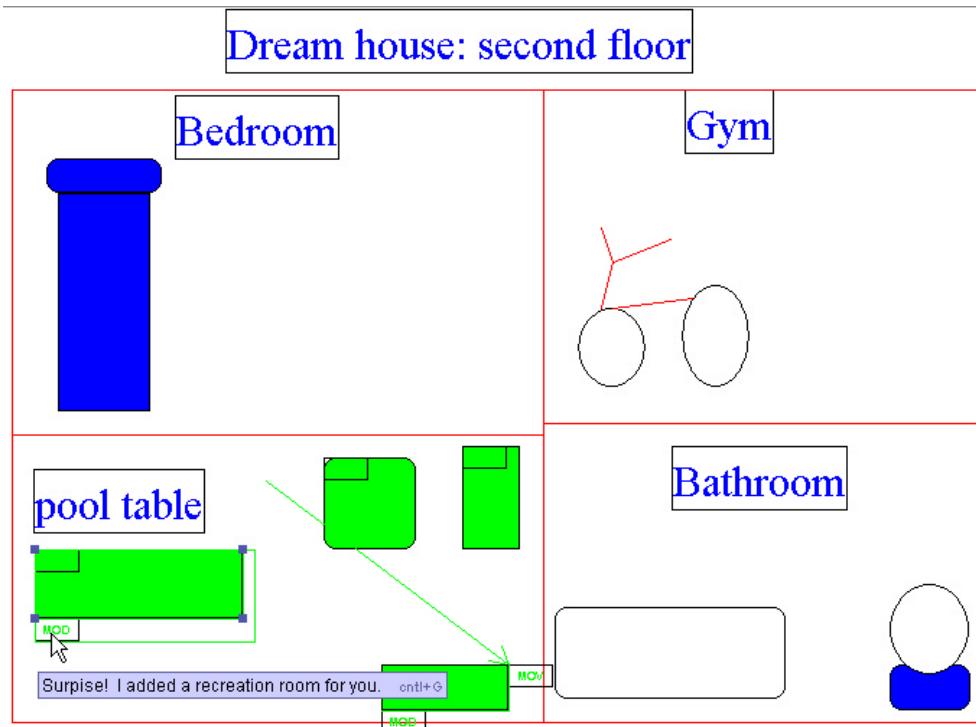


Figure 6.5: Changes to the second floor and the documentation of a specific change.

In summary, this scenario shows the major features of PastDraw that can be used to track changes:

- Overview of changes in a group of related diagrams (thumbnails in the project overview).
- Overview of changes for a single document (document overview).
- Color-coding to provide a quick view of who made changes to which objects.
- Different mechanisms for displaying detailed information about changes animated replays and text labels augmented with graphical cues: movement arrows, after-images of resizes and the resizing arrows, and the faded deletion indicators.
- The ability to see changes represented at different levels of detail: text labels supplemented with color and graphical cues that provide different levels of representation for changes.
- The ability to filter changes by different criteria: the person who made the change, the type of object changed, and the type of change that occurred.
- The ability to ‘accept’ agreed-upon changes so that they are no longer tracked or displayed.

6.2 Analyzing PastDraw’s change tracking

Table 6.1 shows that PastDraw performs an incomplete job of tracking the change history of graphical documents. The informational elements from Chapter 3 not only help to categorize the information but also act as a useful checklist of what should be included. Throughout this section I will be referring to various cells of the second and third columns from this table while I analyze PastDraw’s change tracking. As will be seen much of the focus in PastDraw is an artifact-based action history of changes.

As I indicated in Section 3.3, the action history (ninth row) includes all of the low-level edits in the workspace. It is obvious that this information is of paramount importance when tracking changes as it may be the first and only piece of information that a person will desire when catching up on changes. The types of changes that are tracked by PastDraw include: the addition of new objects, and the deletion, modification and movement of existing objects. The only type of modification that PastDraw tracks is the resizing of objects. Although as shown in the previous section, PastDraw allows for the creation of text objects, changes to text are not tracked. This could be a serious omission depending upon how often that this type of object is needed.

It is likely most people tracking changes to text will typically be interested in seeing changes on a word-by-word basis. However, as illustrated with Flexible Diff system (Neuwirth, Chandhok, Kaufer, Erion, Morris and Miller 1992) described in Chapter 2, because the desired level of granularity may vary it is important that PastDraw not only track changes to text but also do so at different levels of detail.

Category of questions	Informational elements	Does PastDraw track the element?
Where	Gaze history	○
	Location history	○
	Edit history	●
Who	Presence history	○
	Identity	○
	Authorship history	○
	Readership history	○
What	Action History	●
How	Process history	○
	Outcome history	○
When	Event history	●
Why	Cognitive history	○
	Motivational history	○

- Indicates that PastDraw does not track changes for this informational element
- Indicates that PastDraw indirectly or only partially tracks changes for this informational element.
- Indicates that PastDraw does track changes for this informational element.

Table 6.1: Summary of PastDraw's change tracking.

Because the informational elements are inter-related, action history provides partial information about other elements, which include: presence, location, readership and gaze history (Rows 5, 3, 8 and 2 respectively). However, the action history only describes these elements in terms of changes made. If we know that changes have been made we can infer that someone has loaded up PastDraw (i.e., someone was present in the workspace) as well as being able to infer some information about where the person was located (the changed documents) and what was ‘read’ or ‘gazed’ upon (the changed objects and the approximate area around them). If no changes have been made then we still have no information about these four elements because people could have simply loaded up PastDraw and looked around without touching anything. Having explicit information about presence history in PastDraw may be useful as a time saving feature. If you know that no one has even loaded any drawings with PastDraw, there is little point in checking for changes. Once one has determined that others have been present, then knowing which documents have been loaded (i.e., the location history) could be useful under certain circumstances. For example, a person may be waiting on another person’s approval of a document. Knowing that the second person has already looked at the diagram and not expressed any objections or made any changes may imply implicit acceptance of the current version of the document. Consequently PastDraw should provide details about presence and location history even when no changes have been made. This level of knowledge should be sufficient – typically it is not the individual elements in a diagram that count but it is their combination (in the form of location history) that matters. Thus, the lack of direct information about gaze and readership history is not of grievous concern.

Information about where changes occurred, the edit history (fourth row), is also indirectly tracked through action history and the fact that changes are tracked on an object-by-object basis (as well as for the entire document). As Figure 6.3 shows, knowing the changes made to each object (action history) indicates the places where edits occurred.

The action history (ninth row) also includes information about the process and outcome history (tenth and eleventh rows respectively). As I described in Section 3.3.4, the process history describes the sequence of changes in which the workspace evolved from its past state to its current state. The outcome history includes a subset of this information, the before and after state of the workspace. There are times that a person would need the outcome and process history in addition to the action history. Simply knowing what changes were made to a diagram (action history) may not provide enough information to determine if the changes are acceptable or not. This person may need some mnemonic cues about what the diagram looked like before to be able to place these changes in context. In such cases, the need for outcome history becomes apparent. Process history may be needed in PastDraw when people have to reconstruct the details of changes made. For example, when two people are brainstorming ideas using the text object in PastDraw information about process history could be needed to see some of the intermediate ideas that they generated. Except for the need to track modifications to textboxes, the current level of change tracking for the process and outcome history appears to be adequate.

I have already discussed how the readership aspect of identity is partially tracked through the action history. PastDraw also tracks the other aspect of identity, authorship history - as individual changes are made (seventh row). Knowledge about authorship may be useful for reconstructing the interaction history of an object or if a person simply wanted to know who he or she should talk to for further information. Awareness of what objects were changed by a specific person is of obvious importance from a person-based view of the workspace. However, the only authorship information that is tracked by PastDraw is the editor's login name, which is chosen by the person the first time that he or she logs into PastDraw. However, because other collaborators may not always associate the correct login name with another person this level of information may be insufficient for tracking authorship. Ideally authorship information should include additional details such as the editor's full name, email address and phone number that not only helps to identify the person behind a change but also facilitate getting in touch with

him or her. PastDraw should also provide this type of authorship information to make authorship information more useful.

Event history (twelfth row) is tracked in PastDraw both in terms of the exact timing of events (date and time) as well as the order in which they have occurred. Although it is more likely that a person would be interested in having only a rough indication of when a change occurred rather than the exact point in time in and of itself this level of precision is not problematic. It's irrelevant how much detail is stored so long as the system can display the appropriate level of detail when needed and this is how PastDraw fails to provide change awareness for this informational element (refer to Section 6.3).

PastDraw does not automatically track ‘why’ changes were made because as indicated in Section 3.3.6 this may be extremely difficult for an application to do. Consequently PastDraw only tracks the cognitive history (thirteenth row) by allowing users to manually enter their reasons for the changes that they made. Because manual methods of documenting changes may incomplete, incorrect or even omitted altogether, adding even a rudimentary form of automatic documentation to PastDraw may be prove to be useful.

Using the framework from Chapter 3, I have determined what change information is and is not tracked by PastDraw. In some cases, the absence of an informational element does not have a large effect on change tracking (within the context of simple graphical editor). In other cases, the lack of the information can seriously impair a person’s ability to track changes. PastDraw needs to be able track information on location and presence history independent of action history. Additional details, such as contact information, should be provided for authorship history. Action, process and outcome history should track changes made to text at multiple levels of detail and there is a need for the automatic tracking of cognitive history.

6.3 Analyzing PastDraw's display of changes and suggested improvements

In the first subsection, I analyze the specific strengths and weaknesses of PastDraw's display mechanisms by determining what change information (in the form of informational elements) is and is not displayed by each mechanism. In the subsection that follows, I then classify each of these mechanisms according to the display dimensions that I laid out in Section 4.1. This provides insight about ability of a display mechanism for showing change awareness information in this type of drawing application. In the last subsection, I analyze the filtering mechanisms employed by PastDraw.

6.3.1 Critiquing PastDraw's display mechanisms

As indicated in Section 6.2, PastDraw does not track all of the potentially important informational elements. To make matters worse, none of PastDraw's display mechanisms fully display this reduced set of informational elements or the mechanisms will only partially display an element (Table 6.2). In the remainder of this subsection I will be referring to the different cells of this table as I analyze each of PastDraw's display mechanisms.

As Figure 6.2 showed, PastDraw has three main mechanisms for displaying changes: animations, the short text labels and the overviews. One of the overviews shows a high-level view of the diagram (document overview) that is currently displayed in the main view and the other shows the changes made to a collection of diagrams (project overview). All of these mechanisms are augmented by applying color to changed objects. This was done because study participants from our early pilot test of prototype display mechanisms (described in Chapter 5) liked the quick overview of changes that color provided. In addition to this, the text labels were augmented with additional mechanisms: graphical cues (e.g., movement arrows) and text documentation that

provides a more extensive description of changes. In the remainder of this section I will critique each of these mechanisms in turn.

Display mechanism	Categories of questions							
	Where	Who		What	How		When	Why
	Edit history	Identity	Author-ship history	Action history	Process history	Outcome history	Event history	Cognitive history
Animations	●	○	○	●	●	○	●	○
Document overview (animations)	●	○	○	●	●	○	●	○
Text labels	●	○	○	●	○	○	○	○
Document overview (text labels)	●	●	●	●	○	○	○	○
Text doc.'s	○	●	●	●	○	○	●	●
Color	●	●	●	○	○	○	○	○
Graphical cues	●	○	○	●	○	●	○	○
Project overview	●	●	●	○	○	○	○	○

- Indicates that the display mechanism does not display changes for this informational element
- Indicates that the display mechanism only partially displays changes for this informational element.
- Indicates that PastDraw does display changes for this informational element.

Table 6.2: Summarizing PastDraw's display of informational elements.

As seen in Row 2 of Table 6.2, the animations employed in PastDraw focus on the edits made to diagrams: what they were (action history), where they occurred (edit history) and the process of change that the workspace underwent (process history). The order of changes can be determined by the order of the animations (partial event history).

The main draw back of the animated replays was hinted at in Section 6.1: it can be difficult to remember specific details of a sequence of changes, such as order, if many changes have occurred. This omission is compounded by the fact that there is no time-based filtering mechanism available (discussed further in Section 6.3.3) or the ability to see the compressed view of changes in the form of outcome history. Instead the viewer has to watch the full sequence of changes in the form of process history. In the task example portrayed in Section 6.1, the ability to playback only the changes that took place

during the night would have been useful for James. Also, no timing information for ‘when’ is provided during the animations, which would have also been useful for James, because then he could have at least noted the point at which the animations started showing the changes made during the previous night. While it was possible in the task example (Section 6.1) for James to filter out his own changes or the changes of others, PastDraw does not indicate the person who was responsible for each change as the replay is occurring. Finally no information is provided about ‘why’ changes were made.

Providing either a literal presentation of ‘who’ (e.g., a portrait or image of the person making the change) or a symbolic presentation (e.g., having the object change colors during the animation to match the color of the person who made the change) would provide useful information. Viewers would know who to ask for further information about the changes made to an object. Another useful additional would be to augment PastDraw’s animations with some sort of high speed ‘skimming’ or compression mechanism that allows the user to quickly review overall details of changes and then view at normal speed the details of interesting changes. Finally the animations could be supplemented to display ‘why’ changes were made by showing text documentation during the animations.

Row 3 shows that when the display mechanism is set to ‘animations’ that the document overview shows much the same change information as the main view only in miniaturized form. The advantage to having the document overview is that while parts of the document may be obscured in the main view, the entire document is shown in the document overview. This should reduce the possibility that important changes will be missed.

The potentially rich encoding scheme of text works well for displaying abstract information (such as ‘why’) or precise information (such as ‘when’). However, for both text labels and text documentation it is difficult to represent some types of changes (e.g., spatial movements or the physical resizing of objects). It may be useful to exploit the

ability of text to represent abstract information and combine it with a mechanism that better represents spatial changes (such as animations).

Text labels (Row 4) provide only a brief overview of changes: what actions occurred and where they occurred. For some types of specialized structured drawing applications (e.g., UML editors) where more complex changes can occur, the brief text descriptions may prove to be inadequate in representing some types of changes. However, for simple drawing tools such as PastDraw, they appear to be more than sufficient.

It is obvious from the table that this mechanism is missing information on ‘who’, ‘how’, ‘when’ and ‘why’. To find out additional details about some of these elements, the person must look to the more extensive text documentation and supplementary graphical cues.

When the display in PastDraw is set to ‘text labels’ it can be seen in Row 5, that the document overview shows information on where (in the form of edit history), what (in the form of action history) and partial authorship information. This is because the document overview combines the use of color (used to fill in the changed objects) and position (the position of the miniaturized change labels shown in the overview indicates the changes made to an object). The use of position is illustrated in Figure 6.6, which shows the text labels in the main view (a) and in the document overview (b).

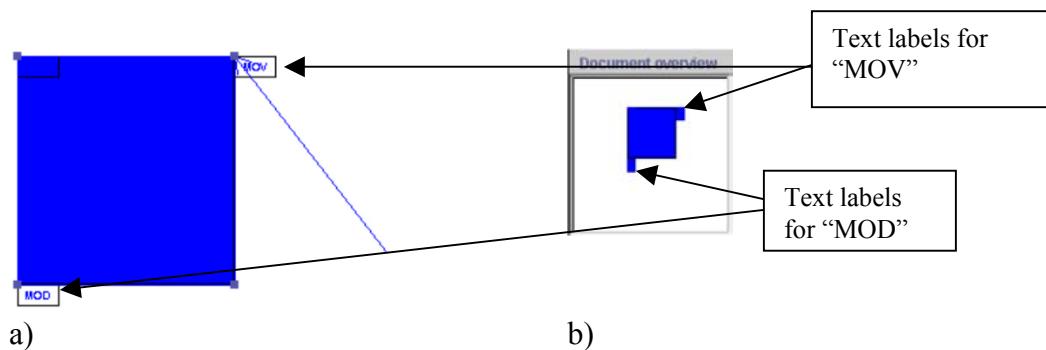


Figure 6.6: Text labels in the main view (a) and the document overview (b).

What is missing from the document overview is information regarding ‘how’ changes occurred, ‘when’ they were made and ‘why’ they occurred. Color fading

(implemented through different levels of saturation and value) could provide approximate timing information as well a general impression about order. The further back in time that an event occurred, the greater the fading. Because color communicates some of the information that is missing from animations, text labels and text documentation it compliments these mechanisms nicely. The absence of the other informational elements from the document overview should not be of grievous concern since this view is only meant to provide a rough idea of the changes that occurred to a diagram. Viewers can look to the main view to see the additional details.

The text documentation shown in Row 6 (Table 6.1) displays automatically generated messages describing changes and documentation explicitly added by users. It provides information about: who made the changes (authorship), what changes were made (action history), when those changes were made (partial event history in the form of timing) and is the only mechanism that can represent the cognitive history ('why') behind changes. Although reading the documentation requires conscious effort, this is okay because color and the text labels provides the quick overview of changes while text documentation provides the additional details only on-demand.

What is missing from this display mechanism is information on 'where' changes were made (edit history) and 'how' things changed (process and outcome history). The main reason for this absence is due to the drawback inherent of text (described previously) – often representing information about these categories of questions involves the representation of spatial information. As indicated earlier, it may be beneficial to combine text documentation with a mechanism (such as the animated replays) that better represents this type of information.

We can see in the Row 7 that color not only indicates where changes occurred but also partial information on who made changes. Objects in PastDraw are filled in with the color of the last person who changed it. For the text labels, the text is also filled with the color of the last person who made that particular change.

However information about several categories of change awareness questions are missing from the color mechanism which include: ‘what’, ‘how’, ‘when’ and ‘why’ changes were made. PastDraw should be augmented so that color can also be mapped to some of the other categories of change awareness questions. Also, color fading (described previously) could have also have been exploited in PastDraw to show an approximate event history.

Graphical cues (eighth row) were used to augment text labels in PastDraw by providing additional details on changes (movements and resizing type of modifications) as well as deletions. These cues not only provide details regarding ‘where’ and ‘what’ changes occurred, they are the only mechanism in PastDraw that display the outcome history (Column seven).

It must be noted, however, that the graphical cues do not display change information for the addition of new objects to a diagram in PastDraw – the text label (“ADD”) was deemed to be sufficient in this case. Also missing is information on the ‘who’, ‘when’ and ‘why’ categories of change awareness questions as well as the process history of ‘how’. Because all the information missing from the graphical cues, save process history, is provided by either color or text documentation these mechanisms work well in conjunction with each other. Although, some of the missing information on process history could be provided on demand (e.g., showing all intermediate locations that an object occupied as it was moved rather than just the start and end points) it is rare that such level of detail would be required.

As mentioned in Section 6.1, the project overview (ninth row) consists of thumbnail images of all the documents in a project.¹ These thumbnails are augmented by color to show information about who made changes and where they occurred. Without the color cues, the project overview only shows information about the current state of documents

¹ It was assumed in PastDraw that all the documents saved together in a disk directory would be all part of an associated collaborative project.

and how many changes occurred to each document (through the bar graph beside each thumbnail).

As can be seen in the table, a great deal of change information is not represented with this mechanism ('what', 'how', 'when' and 'why'). Employing the color fading mechanism (described earlier) will allow for partial information regarding 'when' changes were made. As indicated earlier, it may be possible to allow the user to change the color mapping to other categories of change awareness questions (such as 'what'). The absence of 'how' and 'why' information should not be a large problem because the project overview is meant only to provide a general idea about changes made to all documents.

In addition to these missing informational elements there are a few weaknesses that are specific to this implementation of a project overview. The overviews are static images rather than real-time views of diagrams so that the change filters (Figure 6.2) will not work for the overview. Also, new thumbnails of documents are not created until a document is saved.

In addition to the drawbacks unique to each mechanism, there is no ability for a user of PastDraw to find out about changes without loading up the system. In the task example, Jenn needed to leave James a note before he was aware that any of the diagrams had changed. If changes are infrequent, a simple notification system such as email may be useful, or if changes occur often then the variation of the focus and context technique, illustrated in Figure 4.6, may be employed instead.

6.3.2. Classifying PastDraw's display mechanisms according to the display dimensions

In addition to analyzing each display mechanism according the informational elements that it displays we can gain a general insight about the utility of each mechanism by classifying them according to the display dimensions described in Section 4.1. Figure 6.7 shows how the mechanisms are categorized by dimension.

- 1) Animated replays are a literal, situated and passing type of display.

- 2) Text labels and color are symbolic, situated and permanent types of display mechanisms.
- 3) The document overview can either be a literal, separate and passing display (animations) or a symbolic, separate and permanent display (text labels).
- 4) Text documentation is a symbolic, situated and passing type of display.
- 5) The graphical cues are a symbolic and permanent display that can either be situated (e.g., Figure 6.3, pool table) or separate (e.g., Figure 6.3, resizing of the TV).
- 6) The project overview is a literal, separate and permanent display. This is because it consists of actual thumbnail images of the documents in the project. (The use of color to supplement these image captures is of course an example of a symbolic, separate and permanent display).

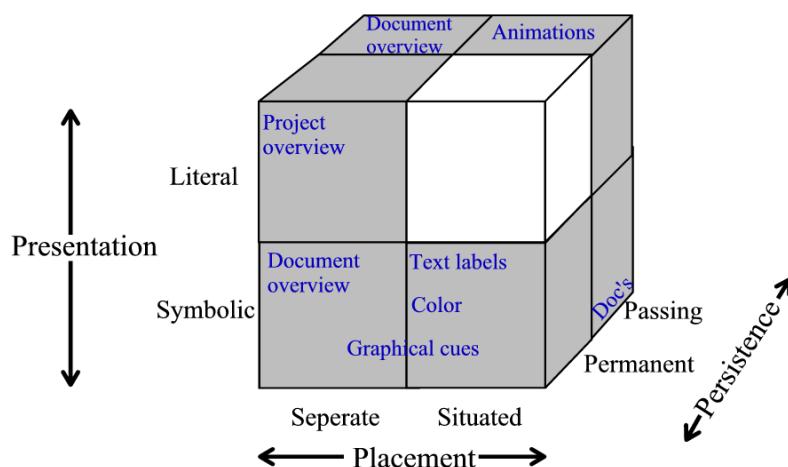


Figure 6.7: Classifying the display mechanisms employed by PastDraw.

Literal presentation, passing persistence (document overview, animated replay): As mentioned in the previous subsection, the animated replay focuses on the edit, action and process history. In terms of the display dimensions, it was seen in the task example that literal presentations make individual changes easy to interpret although getting an overview of how changes relate may be harder if many changes are displayed in sequence.

However, Table 6.2 indicates that the animations in PastDraw do not fully display the informational elements for ‘who’, ‘how’, ‘when’ and ‘why’. As will be seen in the

following discussion the reason for these omissions is partially due to the literal nature of the animations.

Having an approximate idea of ‘when’ changes occurred would be useful. Since it may be difficult to communicate this information in a literal form, the symbolic mechanism recommended in Section 6.3.1 (color) may be used to supplement the animations by providing a rough approximation of the event history. Should greater timing information be required another symbolic mechanism (text) can be used to provide additional details as desired.

Similar to ‘when’, it is difficult to display information about ‘why’ with a literal presentation. Again, it may be necessary to combine it with a symbolic presentation such as text.

As seen in the task example, the passing nature of some representations made it difficult to get an overall picture of changes. James found it difficult to relate all the individual changes together with animations because, aside from the (permanent) color indicators, he couldn’t see any of the change information simultaneously. Providing increased control (frame rate, replay speed, having fast forward and review controls) over the animations and allowing people to ‘bookmark’ key parts of the replay may serve to offset the problems of the ‘passing’ level of persistence.

In summary PastDraw’s literal presentations can be augmented with symbolic presentations such as color and text to display missing informational elements. Providing a greater degree of control over the presentation may offset the passing nature of the representation.

Symbolic presentation, permanent persistence (document overview, text labels, color, graphical cues): As seen in the task example, the advantage of using symbolic and permanent mechanisms is that James was provided with an abstracted and overall view of changes allowing him to associate related events.

However, as noted in Section 6.3.1, it is difficult to communicate some types of information, such as spatial changes, using certain types of symbolic presentations such as text. Consequently it may sometimes be necessary to augment the mechanisms that employ a symbolic presentation with a mechanism that uses a more literal presentation (such as animations).

Separate placement (project overview, graphical cues, document overview): James did not have any problem with the separate placement of changes in the task example. However, Gutwin (1997) indicated that people might have trouble relating workspace awareness information to the workspace with separate placement. Our study participants (Chapter 5) sometimes indicated how annoying gaze shifting was when the placement is separate. In drawing tools such as PastDraw, if many changes occur over time it may be difficult to relate ‘separate’ change display mechanisms, such as the graphical cues, to the object that was changed (Figure 6.8). Although it is possible to determine what resize indicator belongs to what object in Figure 6.8, it requires a fair amount of work to do so and there are only three objects in the diagram. With larger and more volatile diagrams, relating the change indicators to the changed objects becomes more difficult.

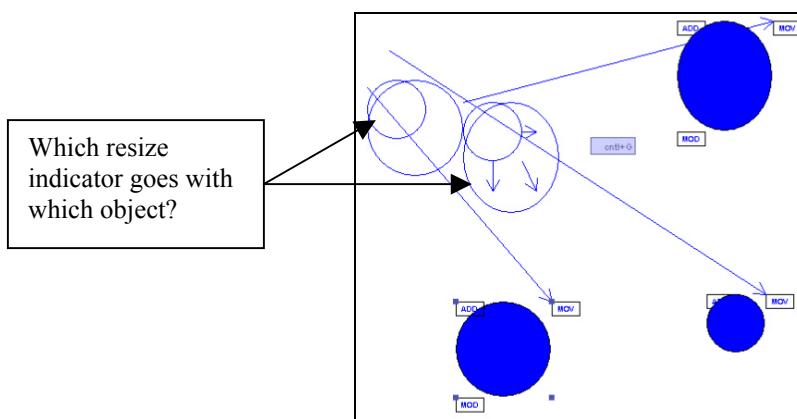


Figure 6.8: Separately placed change mechanisms can be hard to relate to the changed object.

Passing persistence (animated replay, text documentation): The passing level of persistence was shown to be a problem of the animated replays. This was because *all* changes had only a passing persistence with this mechanism. In contrast, the text

documentation was used only to supplement the permanent text labels by providing more detailed change information so its passing level of persistence was not a problem.

6.3.3 Analyzing PastDraw's filtering mechanisms

As seen in Figure 6.2, PastDraw allows users to filter changes according to three categories: 1) by the person who made the change (person-based) 2) by the type of object changed (artifact-based), and 3) by the type of change that occurred (filtering by 'what').

The task example indicates that the person-based filter can be useful, if for nothing else, to screen out the changes that you just made. The value of object-based and action-based is more suspect as we doubt users of PastDraw will differentiate between the objects that can be created and the changes that can be done to them.

As mentioned in Section 6.3.1, what is glaringly missing from the basic change filters is the ability to filter changes according to 'when' it occurred. As I described in Section 3.3.5, 'when' a change occurred can influence the perceived importance of the change. The value of time-based filtering for PastDraw was further illustrated in the task example when James wanted to see only the changes that were made by his wife during the night but could not. He had to see them all.

In the scenario one reoccurring problem was the clutter introduced by irrelevant and/or exploratory changes. Yet, PastDraw had no way of filtering these out. Perhaps semantic filters could be added to automatically screen changes that are deemed as irrelevant. For example, information about a change should not be tracked if the change is immediately reversed. As well these filters could collapse multiple low-level events into higher-level changes that are more meaningful (described in Section 4.4.2). For example, the two separate changes associated with the pool table, a deletion from the first floor (Figure 6.3) and addition to the second floor (Figure 6.5) could be combined into one change, depicting a movement from one part of the floor plan to another.

PastDraw presents multiple hierarchical views of diagrams² through the project overview, the document overview and the main view (Figure 6.2). This three views lend themselves naturally to hierarchical filtering which is partially implemented in PastDraw. As mentioned previously in Section 6.1, the static thumbnail images in the project overview currently show only snapshots of the current state of documents as they are saved. It should instead be a real time view so that people are provided with more updated change information but also so that the change filters can be employed in this overview. The document overview currently provides better hierarchical filtering. It shows a real-time miniaturized view of the current document and a high-level overview of changes (i.e., some of the informational elements) to all the objects in the current diagram. The user can then look at the main view to get additional details.

6.4 Overall analysis and redesign of PastDraw

It was evident from the previous sections that PastDraw's support for change awareness is somewhat lacking. In Section 6.2, by using the framework (from Chapter 3) as a guide, I showed that PastDraw tracked only a subset of the important change information. Next in Section 6.3, I illustrated how none of PastDraw's display mechanisms were able to completely display even this reduced set. Either elements were missing or they were only partially or indirectly displayed by particular mechanisms. In the remainder of this section I show how the framework and display concepts can be applied in the redesign of a structured drawing tool such as PastDraw.

To begin with, users of PastDraw have no idea if anyone else has even run the application or not. As I mentioned in Section 3.3 and 4.2.3 there is little point in checking for changes if no one has even entered the workspace. A simple desktop notification system can be added to PastDraw to let users know that something has changed (Figure 6.9). Users can choose to ignore the indicator, banish it or probe for

² It must be noted that the project overview and the document overview are still a work-in-progress and do not always work properly. The thumbnails presented in the project radar are sometimes blurred and the aspect ratio of objects in the document radar is sometimes incorrect.

further details by loading up PastDraw. If necessary, this indicator can be made more noticeable by having it flash whenever a change occurs.

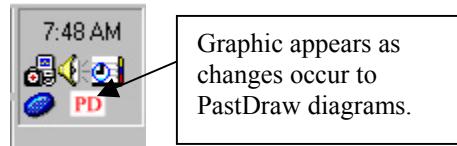


Figure 6.9: Notifying users of changes outside of PastDraw (mockup).

In addition, PastDraw neglects tracking many important information elements. No information is retained about a person's location in the project (i.e., the particular diagrams that were loaded up). Also, while the editor's login name was stored as changes were made, there are situations in which more details would be needed. In Section 6.2, I provided examples of when these informational elements may be useful. Finally, there may sometimes be a need to determine the reasons behind a change. While it may be useful for an application to allow editors to add their own documentation, this does not imply that editors will take advantage of this feature. Even a rudimentary form of automated documentation may be useful. The redesign of PastDraw should account for these missing elements so that they will be tracked by the system.

In Section 6.3 we saw how PastDraw's deficient change tracking was made worse by the fact that each display mechanism showed only a subset of the tracked information. This was partly due to the limitations of the particular octant of display dimension (Section 4.1) that the mechanism represented. While literal presentations were the clearest in presenting some types of changes, such as 'how' things changed, it was difficult to present abstract information (e.g., why changes occurred or when they were made). Some of the symbolic mechanisms, such as text, could be used to offset the weaknesses of the literal mechanisms and vice versa.

For a drawing application such as PastDraw, the high level view of changes provided by the project overview can be important when there are many related diagrams. In this view, a mechanism that roughly shows change information at a glance, such as color, is appropriate. The use of different color saturations and values, described

in Section 6.3.1, can represent the event history. As previously indicated the project overview should consist of real-time views of documents rather than static images so that the change filters can be employed in this view and to allow for better hierarchical filtering of changes.

When a person wishes to probe for further details, a storyboard (simulated in Figure 6.10, bottom) can provide an easily understandable literal view of changes while avoiding the animation's problems of passing persistence. Panels start out with a coarse level of granularity: the far left panel shows the last time that the viewer loaded up the document (in this case when the document was first created) while the panel on the far right shows the current state of the document. A new panel is created for each significant event i.e., every time that someone loads up or closes ('logs out') a document. This allows a person to track when others have been present in the workspace as well as which documents have been viewed by others. The events associated with the creation of each panel are ordered and dated showing the event history of changes made. The chronological overview of changes was one of the features of storyboarding that test participants (Chapter 5) liked about this mechanism. If desired, the detail level of timing information can be increased to indicate the exact time that an event occurred.

If even further details are required then the user can see all of the intervening details between these panels by 'stretching' out the storyboard display. One thumbnail in Figure 6.10 'Roger login, Sept. 16, 2001' is outlined to indicate that it has been selected. Manipulating the slider to the right of the thumbnails allows users to 'drill-down' and to probe for further details on the changes that took place just before and after the selected panel (Figure 6.11). At the detail level shown in the figure, a new panel is created for five changes that were made to a document. Users can continue probing for increasing levels of detail until a panel is shown for every change that occurred. At all levels of details users can request an animated replay of changes that plays back all the intervening changes between panels. This combination provides users with the persistent before and after view of changes that test participants (Chapter 5) said that they wanted added to the

object animations. Also it deals with the persistency problems inherent in the replay mechanism that we discovered in our study.

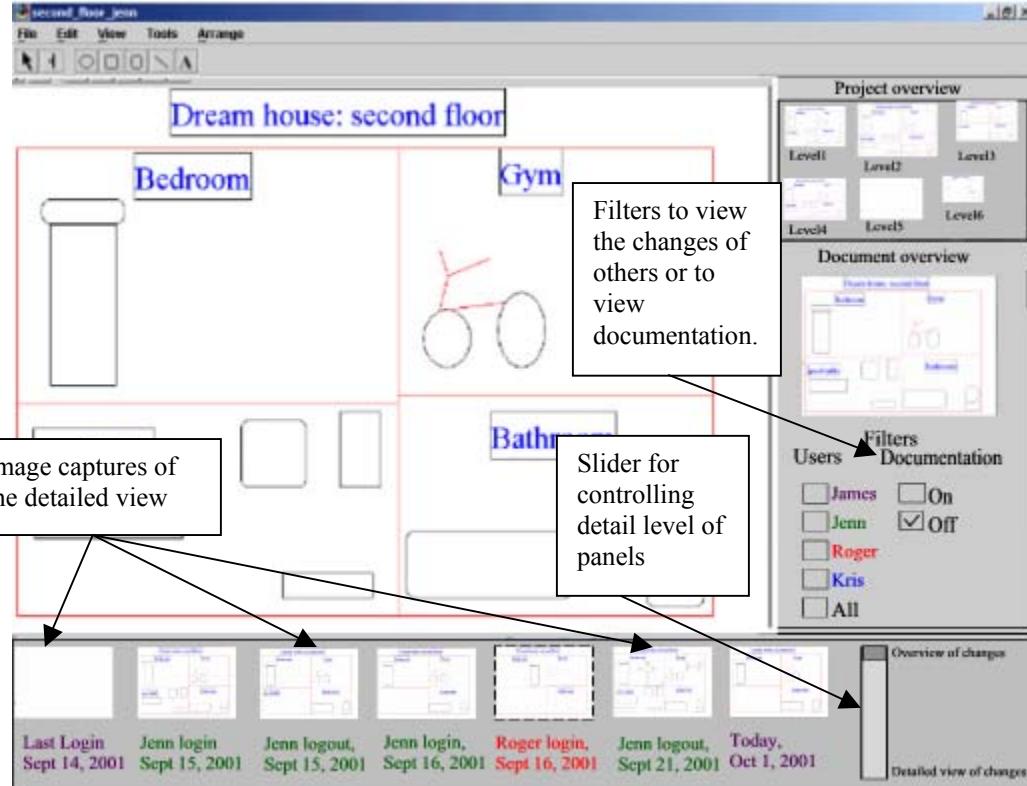


Figure 6.10: The new and improved PastDraw (mockup).



Figure 6.11: Users can increase the granularity of panels to probe for further details (mockup).

Some of the less useful filters (by type of object and by type of change) have been removed from PastDraw. Users can still filter by person. In addition, they can view documentation on a diagram-by-diagram basis rather than on per-change basis. To avoid clutter, users can filter the documentation (and changes) on a per-person basis or turn it off altogether.

The reduction in the number of filters reduces the competition for screen real estate and thus allows the project overview (Figure 6.10, top right) to be made larger to show more diagrams in the overview. These miniaturized views show changes in real-time rather than static thumbnail images. The number of changes made to the document determines the size of each document in the project overview. The greater the number of changes that have occurred to a document, the larger will be its representation in the overview. To further enhance the focus and context effect and to economize on space, only diagrams that have been changed will be shown in the project overview.

The document overview in the redesigned version of PastDraw is almost unchanged save for being made larger and positioned in a different place (Figure 6.10, center right).

Semantic filters can be employed to avoid creating panels for irrelevant small changes. The ‘zooming’ aspect of storyboards allows low-level sequential changes to be combined. The display of the panels will also be controlled by the basic filters. For example, as users toggle the filter controls to block/unblock the changes of other users, panels will appear and disappear. The storyboards allow people to start with a before and after view of changes (outcome history) and then probe for further details as they explore the process history. As already mentioned, animations can be used to provide explicit information about the changes between panels. Although only a portion of the overview of changes and events remains visible at a given time, this is offset by the ability to adjust the granularity between panels, allowing users to change the scope of the overview. Users can see detailed information about changes in the detailed main view by clicking on a panel. The ability to adjust the granularity level deals with problems found in early storyboard prototypes (Chapter 5) where people would sometimes lose the flow of events going from one panel to another because the granularity level was too coarse.

6.5 Discussion of the critique

In the beginning of this chapter I described the difficulties that I encountered when trying to develop PastDraw. The principles described in the framework and display concepts

may seem to be obvious after they have already been formulated in this thesis. However, as seen in much of the discussion in this chapter, I fell into quite a few pitfalls without it. I focused heavily on implementation details because at the time I did not have a clear understanding of what were the important concepts for change awareness.

I initially based my implementation on intuitions on what I felt was a good design as well as utilizing a portion of Gutwin's (1997) framework for workspace awareness. But, as I already indicated in Chapter 3, there were many additional concepts that needed to be elaborated. These concepts formed the basis of my framework for change awareness. As I began describing the important concepts for displaying change awareness information, I drew upon the work of several researchers in information visualization: Card, MacKinlay and Shneiderman (1999); Carpendale (2001); Kurlander and Feiner (1991, 1993); Havre, Hetzler and Nowell (2000); Eick, Steffen and Sumner (1992); Furnas (1981); Apperley and Spence (1982); Spence and Apperley (1982) and as well the work of Bertin (1967).

As shown in Section 6.2 and 6.3.1, the informational elements from the framework can be used to understand the strengths and weaknesses of a change awareness implementation. In Section 6.2, the framework provided a way of classifying and listing what information was and was not tracked by PastDraw. Overall, the information tracked was appropriate to a simple drawing tool although some useful information was either missing or only partially tracked. Section 6.3.1 showed that the main problem with PastDraw's change awareness support was that each display mechanism was only able to show a subset of the information that was tracked. Part of the reason why some elements could not be displayed by a particular mechanism was due to the inherent limitations of the display octant (detailed in Section 4.1). Determining what information that was present or absent for different mechanisms allowed us to see how combining some of the mechanisms could offset their individual weaknesses thus improving PastDraw's change awareness support.

In Section 6.3.2, I then classified the display mechanisms according to the display dimensions to show how this would provide general insight into each of PastDraw's display mechanisms. In addition, the process provided us with an indication of the strengths and weaknesses of future display mechanisms from the same octant for drawing tools similar to PastDraw. The latter information would prove to be especially useful later, in Section 6.4 during the redesign of PastDraw. In Section 6.3.3, it was found that while some of PastDraw's filtering mechanisms (i.e., filter by person) could be useful in a structured drawing application the value of other filters (i.e., filter by the type of object or the type of change) are more suspect.

I then applied the insight gained in Section 6.2, and 6.3 in the redesign of PastDraw in Section 6.4 in order to demonstrate how the framework and display concepts could be used as a high-level design guide. The classification of elements that were tracked and displayed by PastDraw indicated to me what information was important for this type of application and needed to be considered in the redesign. The display dimensions described different ways of representing change information and how they could be combined in the redesigned system.

Chapter 7

Conclusion

In this chapter, I describe how the four research objectives laid out in Chapter 1 were met in this thesis and what my research contributions were. I end by providing some ideas for future areas of research into change awareness.

7.1 Research objectives and contributions

My primary goal in this thesis was to describe the concept of change awareness in detail and what support must be provided by software for dealing with it. This main goal is comprised of four sub-goals.

- 1) To explore ideas in change awareness and to gain a hands-on understanding of it. I do this by extending an existing graphical 2D structured drawing program to track and display change awareness information.
- 2) Produce a conceptual framework for understanding change awareness in 2D graphical collaborative workspaces, where multiple collaborators interact over the space at different times. I do this by expanding Gutwin's framework on workspace awareness (Gutwin 1997), which focused on investigating real time interactions, to include in detail how people can stay aware of changes over time.
- 3) Develop a set of basic concepts to guide the display of change awareness information. This includes the visual representation of change information as well as mechanisms for filtering this information. I do this by applying previous research into information visualization (Card, MacKinlay and Shneiderman 1999) as well as describing ideas that are unique to change awareness.
- 4) Demonstrate the value of the framework and display concepts. I do this by critiquing the system (implemented in the first goal) according to the criteria laid out in the

framework (developed in the second goal) and the display concepts (developed in the third goal).

The ways in which I have met the goals laid out in my thesis are described below.

1) Implement a change awareness support system. In Section 6.1, I describe how I augmented an existing structured drawing application with change awareness to create PastDraw. The features of PastDraw that support change awareness include: multiple hierarchical views of changes through the project and document overview as well through the main view; different mechanisms for displaying changes with varying levels of detail; the ability to filter changes according to different criteria; the ability to accept agreed upon changes.

2) Produce a conceptual framework for change awareness. In Chapter 3, I specified the fundamental categories of questions ('where', 'who', 'what', 'how', 'when', and 'why') that may be asked by a person tracking changes in a workspace. These categories of awareness questions are based on Gutwin's (1997) framework for workspace awareness, which I adapted and expanded specifically for change awareness. For each of the six categories, I described in detail a set of informational elements that provide the answer to that category of questions. Although I focused on 2D graphical workspaces in this thesis, these informational elements are relevant to other types of workspaces (e.g., text based systems) as well. Finally, in this framework, I specified three different ways in which individuals may perceive the workspace: artifact-based, person-based, and the workspace-based view. The particular view of the workspace that a person holds will impact on the type of information that a person is interested in and the way in which he or she queries for that information.

3) Develop a set of basic concepts to guide the display of change awareness information. I described in Chapter 4 some of the important considerations for displaying change awareness information. These considerations include: the crucial dimensions for displaying the information (placement, presentation and persistence); ways of reducing the acquisition and interpretation costs of information; mechanisms

for displaying the information elements; and information filtering to avoid overload. The display mechanisms include: the visual variables first described by Bertin (1967) later extended by Carpendale (2001), storyboarding (Kurlander and Feiner 1991; 1993) and text.

4) Demonstrate the value of the framework and the display concepts by critiquing

PastDraw. In Chapter 6, I showed how the framework and display concepts can be used to both analyze an existing change awareness tool, and as high-level design guides for future system redesign. The categories of change awareness questions were used to assess and understand the strengths and weaknesses of PastDraw's tracking and display of changes by determining what was present and what was missing. While it appeared that PastDraw at least partially tracked most of the important information, individually each of its display mechanisms were found to be lacking. Finally, I applied the framework and display concepts in the redesign of PastDraw. Again, the informational elements guided me in determining what information should be tracked and displayed. The display dimensions described how different mechanisms could be combined in order to offset each other's weaknesses.

7.2 Future work

In this thesis, I focused on change awareness in the general domain of 2D graphical workspaces that contain distinct graphical objects. What is needed is the application of the framework for change awareness and the display considerations to a specific task domain. Ideally, this domain should have a strong set of semantics that not only constrain the set of meaningful changes that can be made but also ascribe a strong meaning to these events. The benefit of domain-specific semantics is that these rules can be used to generate an extended set of informational elements unique to that domain which may increase their value to the people who are tracking changes. In addition, the semantics can be used to develop a useful set of rules for filtering irrelevant changes. Because the informational elements and filtering are tied to the semantics of the domain,

it will be more likely the pertinent information is retained while the irrelevant information is filtered out.

There has already been some research conducted into information filtering by Dellen (1999). She developed a general mechanism for describing semantic filters in arbitrary task domains. She implemented this mechanism in the process modeling tool MILOS (Dellen 1999). However, while she explored in detail the issue of filtering she did not account for the different ways that change information may be represented and instead settled on a notification based system which may not always be the most effective mechanism for displaying change awareness information.

So as well as determining what information is needed to track changes in specific task domains and exploring different ways filtering these changes, future research must also consider the means of displaying this information. Again, the semantics of this area may be used as a guide for selecting the most effective method of representation.

Also what is needed is the complete development of a change awareness tool (such as a revised version of PastDraw) according the principles laid out in the framework and display concepts. User testing can then be conducted to see how well the framework and its application to system design matches actual user demands.

Finally, while this thesis focused exclusively on change awareness in 2D graphical workspaces, future research should be expanded to include other types of graphical domains as well. For example, change awareness in three dimensional CAD/CAM applications may present some unique challenges. While the concepts from my framework can be used to determine what change information should be tracked, the display of this information in 3D requires new issues to be faced (e.g., occlusion causing some changes to be obscured by objects in space).

References

- Allan, G.W. (1997) *What is Configuration Management?* Logistics Spectrum, Vol. 31 No. 1: pp. 15-18.
- Anderson, J.R. (1983) *The Architecture of Cognition*. Harvard University Press (Cambridge MA).
- Apperley, M. and Spence, I. T. R. (1982). *A bifocal display technique for data presentation*. Proceedings of Eurographics '82, pp. 27-43.
- Arango, G., Schoen, E., Pettengill, R. and Hoskins, J. (1993) *The Graft-Host Method for Design Change*. Proceedings of the International Conference on Software Engineering, ICSE '93 (Baltimore, MD), pp. 243 – 254.
- Benford, S., Bowers, J., Fahlen, L., Greenhalgh, C. and Snowdon, D. (1995) *User Embodiment in Collaborative Virtual Environments*. Proceedings of Human Factors in Computing Systems, CHI'95 (Denver Colorado), pp. 242 – 249.
- Berliner, B. (1990) *CVS II: Parallelizing Software Development*. Proceedings of 1990 Winter Usenix Conference (Washington D.C.).
- Bertin, J. (1967) *The Semiology of Graphics: Diagrams, Networks, Maps*. (Berg, W.J. Trans.), University of Wisconsin Press, Translated version 1983.
- Booch, G., Rumbaugh, J. and Jacobson, I. (1999) *The Unified Modeling Language User Guide*, Addison-Wesley.
- British Standards Institution. (1984) Configuration management of computer-based Systems, <http://www.bsi-global.com>.
- Brown, H.B. (1970) *The Clear/Caster System*. Proceedings of 1970 NATO Conference on Software Engineering (Rome Italy).
- Card, S.K., MacKinlay, J.D. and Schneiderman, B. (1999) *Readings in Information Visualization Using Vision to Think*. Morgan Kaufmann Publishers Inc. (San Francisco, CA).
- Card, S.K., Robertson, G.G. and Mackinlay, J.D. (1991) *The Information Visualizer, an Information Workspace*. Proceedings of Human Factors in Computing Systems, CHI'91 (New Orleans, Louisiana), pp. 181 – 188.

- Carpendale, M. S. T. (2001) *Considering Visual Variables as a Basis for Representing Information*. Technical Report No. CPSC 2001-693-16, Department of Computer Science, University of Calgary, (Calgary Canada), December.
- Cavalier, T., Chandhok, R., Morris, J., Kaufer, D. and Neuwirth, C. M. A. (1991) *Visual Design for Collaborative Work: Columns for Commenting and Annotation*. Proceedings of the Twenty-fourth Hawaii International Conference on System Sciences, HICSS-24 (Kalua Kona, Hawaii), pp. 729 – 738.
- Clark, H.H. and Brennan, S.E. (1991) *Grounding in Communication*. In Readings in Groupware and Computer Supported Cooperative Work: Assisting Human-Human Collaboration, Baecker, R.M., Ed., Morgan Kaufmann Publishers Inc., pp. 222 – 233.
- Compaq (1975) *VAX/VMS*. <http://www.openvms.compaq.com/>
- Cross, G.A. (1990) *A Bakhtinian Exploration of Factors Affecting the Collaborative Writing of an Executive Letter of an Annual Report*. Research in the Teaching of English, Vol. 24 No. 2, pp. 173 – 203.
- DEC. (1982) *Code Management System*. Document No. EA-23134-82, Digital Equipment Corporation.
- Dellen, B. (1999) *Change Impact Analysis Support for Software Development Processes*. Ph.D. thesis, Department of Computer Science, University of Kaiserslautern (Kaiserslautern, Germany).
- Dourish, P. and Bly, S. (1992) *Portholes: Supporting Awareness in a Distributed Work Group*. Proceedings of the Conference on Human Factors in Computing Systems, CHI' 92 (Monterey, CA), pp. 541 - 547.
- Eick, S.G., Steffen, J.L. and Sumner, E.E. (1992) *Seesoft—A Tool for Visualizing Line Oriented Software Statistics*. In Readings in Information Visualization, Card, S.K., MacKinlay, J.D. and Shneiderman, B., Ed., Morgan Kaufmann Publishers Inc., pp. 419 – 430.
- Endsley, M. (1995) *Measurement of Situational Awareness in Dynamic Systems*. Human Factors, Vol. 37 No. 1: pp. 65 – 84.
- Fowler, M. and Scott, K. (1997) *UML Distilled: Applying the Standard Object Modeling Language*, Addison Wesley Longman.
- Furnas, G.W. (1986) *Generalized fisheye views*. Proceedings of Human Factors in Computing Systems, CHI '86 (New York, N.Y.) pp. 18-23.

- Gaines, B.R. and Shaw, M.L.G. (1995) *Concept Maps as Hypermedia Components*. International Journal of Human-Computer Studies, Vol. 43 No. 3, pp. 323 – 361.
- Greenberg S. and Roseman, M. (Forthcoming 2001) *Using a Room Metaphor to Ease Transitions in Groupware*. In Beyond Knowledge Management: Sharing Expertise, Ackerman, M., Pipek, V., Wulf, V., (Eds), MIT Press (Cambridge, MA).
- Gutwin, C. (1997) *Workspace Awareness in Real-Time Groupware Environments*. Ph.D. thesis, Department of Computer Science, University of Calgary, (Calgary Canada).
- Gutwin, C., Stark, G. and Greenberg, S. (1995). *Support for Workspace Awareness in Educational Groupware*. Proceedings of the ACM Conference on Computer Supported Collaborative Learning (Bloomington, Indiana), pp 17 – 20.
- Habermann, A.N. (1979) *A Software Development Control System*. Technical Report, Department of Computer Science, Carnegie-Mellon University (Pittsburgh, PA), January.
- Havre, S., Hetzler, B. and Nowell, L. (2000) *Theme River: Visualizing Theme Changes Over Time*. Proceedings of the IEEE Symposium on Information Visualization, InfoVis2000 (Salt Lake City, Utah), pp. 115 – 123.
- Hill, W.C. and Hollan, J.D. (1992) *Edit Wear and Read Wear*. Proceedings of Human Factors in Computing Systems, CHI'92 (Monterey CA), pp. 3 – 9.
- Hunt, J., W. and McIlroy, M.D. (1975) *An Algorithm for Differential File Comparison*. Computing Science Technical Report No. 41, Bell Laboratories.
- IBM (1991) *VisualAge*. <http://www-4.ibm.com/software/ad/vajava/>.
- Kahneman, D. (1973) *Attention and Effort*. Prentice-Hall (Englewood Cliffs, N.J.).
- King Features Syndicate Inc. (2001) *Slylockfox*. <http://www.slylockfox.com>.
- Koffka, K. (1935) *Principles of Gestalt Psychology*. Harcourt-Brace (New York, NY).
- Korf, R.E. (1987) *Macro-Operators: A Weak Method for Learning*. Artificial Intelligence Vol. 27: pp. 35 – 77.
- Kurlander, D. (1993) *Graphical Editing by Example*. Proceedings of Human Factors in Computing Systems, InterChi'93, (Amsterdam, The Netherlands), pp. 529.

- Kurlander, D. and Feiner, S. (1991) *Editable Graphical Histories: The Video*. Proceedings of Human Factors in Computing Systems, CHI'91, (New Orleans Louisiana), pp. 451 – 452.
- Kurlander, D. and Feiner, S. (1993) *A History of Editable Graphical Histories*. In Watch What I Do Programming by Demonstration, Cypher, A., Ed., MIT Press (Cambridge, MA.).
- Larkin, J., McDermott, J., Simon, D. and Simon, H.A. (1980) *Models of Competence in Solving Physics Problems*. Cognitive Science Vol. 4: pp. 317 – 348.
- Larkin, J.H. and Simon, H.A. (1987) *Why a Diagram is (Sometimes) Worth Ten Thousand Words*. Cognitive Science Vol. 11: pp. 65 - 99.
- Lesser, V., R. and Erman, L.D. (1980) *Distributed Interpretation: A Model and Experiment*, IEEE Transactions on Computers Vol. 29 No. 12: pp. 1144 – 1163.
- Luqi. (1990) *A Graph Model for Software Evolution*. IEEE Transactions on Software Engineering, Vol. 16 No. 8: pp. 917 – 927.
- Mackinlay, J.D., Robertson, G.G. and Card, S. K. (1991) *The Perspective Wall: Detail and Context Smoothly Integrated*. Proceedings of Human Factors in Computing Systems, CHI'91 (New Orleans Louisiana), pp. 173 - 179.
- Magnusson, B. and Asklund, U. (1996) *Fine Grained Version Control of Configurations in COOP / Orm*. Proceedings of Symposium on Configuration Management, SCM6 (Berlin, Germany).
- McCaffrey, L. (1998) *Representing Change in Persistent Groupware Environments*. GroupLab report, Department of Computer Science, University of Calgary, (Calgary, Canada). www.cpsc.ucalgary.ca/grouplab/papers/
- McGrath, J. (1994) *Methodology Matters: Doing Research in the Social and Behavioral Sciences*. Readings in Human-Computer Interaction: Toward the Year 2000, Baecker, R., Grudin, J., Buxton, W. and Greenberg, S., Ed., Morgan Kaufmann Publishers Inc. (San Francisco, CA) pp. 152 – 169.
- Microsoft Corporation. (1983) *Word*. <http://www.microsoft.com/office/word/default.htm>.
- Microsoft Corporation. (1992) *Visual SourceSafe*. <http://msdn.microsoft.com/vstudio/>.

- Microsoft Corporation (1995). *PhotoDraw*.
<http://www.microsoft.com/office/photodraw/default.htm>.
- Nachbar, D. (1988) *Spiff—A Program for Making Controlled Approximate Comparisons of Files*. The Proceedings of the Summer 1988 USENIX Conference (San Francisco, CA) pp. 73 – 84.
- Neuwirth, C.M. and Kaufer, D.S. (1989) *The Role of External Representations in the Writing Process: Implications for the Design of Hypertext-based Writing Tools*. Proceedings of Hypertext '89 (Pittsburgh, PA), pp. 319 – 342.
- Neuwirth, C.M., Chandhok, R., Kaufer, D.S., Erion, P., Morris, J. and Miller, D. (1992) *Flexible Diff-ing in a Collaborative Writing System*. Proceedings of Conference on Collaborative Work, CSCW '92 (Toronto Canada), pp. 147 – 154.
- Neuwirth, C., M., Kaufer, D.S., Chandhok, R. and Morris, J., H. (1990) *Issues in the Design of Computer Support for Co-authoring and Commenting*. Proceedings of Conference on Collaborative Work, CSCW'90 (Los Angeles, CA), pp. 183 – 195.
- Nielsen, J. (1993) Usability Engineering, Morgan Kaufmann Publishers Inc (San Francisco, CA).
- Nilsson, N.J. (1980) *Principles of Artificial Intelligence*. Tioga Publishing Co (Paolo Alto, CA).
- Plaisant, C., Milash, B., Rose, A., Widoff, S. and Shneiderman, B. (1996) *Lifelines: Visualizing Personal Histories*. Proceedings of Human Factors in Computing Systems, CHI'96, (Vancouver Canada), pp. 221 - 227.
- Pressman, R.S. (1997) *Software Engineering a Practitioner's Approach Fourth Edition*, McGraw-Hill.
- Rational Software Corporation. (1991) *Rational Rose*.
<http://www.rational.com/products/rose/index.jsp>.
- Resnikoff, H.L. (1989) *The Illusion of Reality*. Springer-Verlag (New York, NY).
- Rochkind, M.J. (1975) *The source code control system*. IEEE Transactions on Software Engineering Vol. 1 No. 4: pp. 364-370.
- Roseman, M. (1996). *Managing Complexity in TeamRooms, a Tcl-Based Internet Groupware Application*. Proceedings of the Tcl/Tk Workshop (Monterey, CA).

- Roseman, M. and Greenberg, S. (1996a). *TeamRooms: Network Places for Collaboration*. Proceedings of Conference on Computer Supported Cooperative Work, CSCW'96 (Cambridge MA), pp. 325 – 333.
- Roseman, M. and Greenberg, S. (1996b). *TeamRooms: Groupware for Shared Electronic Spaces*. Proceedings of Conference on Human Factors in Computing System, ACM SIGCHI'96 (Vancouver, BC), pp. 275 - 276.
- Rumbaugh, J., Jacobson, I. and Booch, G. (1999) *The Unified Modeling Language Reference Manual*. Addison-Wesley.
- Samuels, S.J., and Kamil, M.L. (1984) *Models of the Reading Process*. In the Handbook of Reading Research, D.P. Pearson, Ed., Longman Inc., N.Y., pp. 185 – 224.
- Schaffer, D., Zuo, Z., Greenberg, S., Bartram, L., Dill, J., Dubs, S. and Roseman, M. (1996). *Navigating Hierarchically Clustered Networks Through Fisheye and Full-Zoom Methods*. ACM Transactions on Computer-Human Interaction, Vol. 3 No. 2: p162-188.
- Shneiderman, B. (1983) *Direct Manipulation: A Step Beyond Programming Languages*, IEEE Computer, Vol. 16. pp. 57 – 68.
- Shneiderman, B. (1984) *Dynamic Queries for Visual Information Seeking*, IEEE Software Vol. 11 No. 6: pp. 70 – 77.
- Shneiderman, B. (1996) *The Eyes Have It: A Task by Data Type Taxonomy for Information Visualization*. Proceedings of IEEE Symposium on Visual Languages, VL'96 (Boulder, CO), pp. 336 – 343.
- Sobell, M.G. (1999) *A Practical Guide to Solaris*. Addison-Wesley (Reading, MA).
- Spence, R. and Apperley, M.D., (1982) *Data base navigation: an office environment for the professional*. Behaviour and Information Technology Vol. 1 No. 1, pp.43-54
- Steves, M.P., Morse, E., Gutwin, C. and Greenberg, S. (2001) *A Comparison of Usage Evaluation and Inspection Methods for Assessing Groupware Usability*. Proceedings of the International Conference on Supporting Group Work, Group '01 (Boulder, CO), <http://www.cpsc.ucalgary.ca/grouplab/papers>
- Suchman, L.A. (1987) *Plans and Situated Actions: The Problem of Human Machine Communication*. Cambridge University Press.
- Sun Microsystems Inc. (1996) Java. <http://java.sun.com/j2se/1.3/>.

Tam, J., McCaffrey, L., Maurer, F. and Greenberg, S. (2000) *Change Awareness in Software Engineering Using Two Dimensional Graphical Design and Development Tools*. Technical Report 2000-670-22, Department of Computer Science, University of Calgary (Calgary, Canada),
<http://www.cpsc.ucalgary.ca/grouplab/papers>

TeamWave Software Ltd. (1997) *TeamWave*. <http://www.teamwave.com>.

Thomas, D. and Johnson, K. (1988) Orwell: *A Configuration Management System for Team Programming*. Meyrowitz, N. Ed., Proceedings of Conference on Object-Oriented Programming Systems, Languages and Applications, OOPSLA'88 (San Diego, CA), pp. 135 - 141.

Tichy, W. F. (1991) *RCS – A System for Version Control*. Software – Practice and Experience, Vol. 15 No. 7: pp 637 – 654.

Tigris. (1995) *GEF* (the Graph Editing Framework). <http://gef.tigris.org/index.html>.

Tigris. (1999) *Argo/UML*. <http://argouml.tigris.org/>.

Tigris. (2000) An open source research community, <http://argouml.tigris.org>.

Wroblewski, D., Hill, W.C. and Mccandless, T. (1990) *Attribute-mapped Scroll Bars*, U.S. Patent Applications: Serial No. 07/523,117 filed May 14, Serial No. 07/626, 130 filed Dec 11, 1990.

Appendix A: Usability study materials

A.1 Per-participant procedures

- Be sure the computer is in the proper start state as follows:
- 1280 x 1024 screen resolution.
- Internet Explorer loaded to the index page and set to **full screen**.
- Print four copies of the post-scenario questions, plus 1 copy of the pre/post test questions.
- Greet the participant at the door of the lab so that they do not have to find you. Introduce yourself. Engage in idle chit-chat if the participant looks nervous.
- Seat the participant in front of the computer and sit beside him or her.
- Explain that you will often be reading from this script to be sure that every participant gets the same information.
- Recite the following:

We are doing an ‘interface refinement’ study to determine which type of change display mechanisms work best for describing changes that have happened to UML diagrams. With your help, we hope to generate ideas that could make it into future generations of UML editors.

We will be asking you to interact with a prototype UML editor, study the UML diagrams it shows, and answer some questions about them. You may find that you do not have enough information to answer all the questions; ‘I don’t know’ is a perfectly valid answer. If at any point during the study you wish to leave altogether, that option is always within your rights. The whole procedure should take about an hour to complete, and we are authorized to pay you \$10 for your participation.

Do you have any questions?

- Have the person read and sign the tester consent form.
- Recite the following:

To specific of your eligibility for the study, I / we would appreciate it if you could please answer the following questions...

- Give the pre-test questionnaire to the participant.
- Explain the role that the subject should play:

The class diagrams you are about to see describe a fictitious piece of software to help accountants manage their clients. You are to pretend that you are a developer working on this software. Imagine that you have been away from this software project for a period of time and must now determine what has changed since the last time that you worked on it. You do not have the luxury of having a backup copy of the UML diagrams in order to do a before-after comparison. However the UML editor your

team uses has a sophisticated change management system that provides you with details about the changes that have occurred. You are invited to use these the information provided to help you answer questions about the changes.

- Explain how the testing procedure will go:

I will call up the UML diagram for you. Because this is a prototype, the diagram has only limited interaction capabilities. That is, you can click on certain areas of the diagram and it will respond. After brief exploratory period, I will ask you some questions about the changes that have occurred to the diagram. While answering the questions, interacting with the diagram is allowed (and even encouraged). This procedure will be repeated for four different scenarios.

- Call up the first scenario for this participant. Then let the participant examine the diagram until they are done.
- Ask the scenario questions from the following pages. For questions 2 and 3, write down whether or not the participant correctly identified the changes (as opposed to writing down his literal responses to the question).
- Call up the second scenario for the participant, and repeat.
- After all scenarios have been run, recite the following:

You've now seen the four candidates we're considering using for displaying change information on UML diagrams. We would be interested in your opinions. First, a couple of specific questions:

- Verbally deliver the post-test questions:
- Ask the participant for any additional questions or information he may wish to volunteer:
- Do you have any additional comments or questions about these change display mechanisms, or the task of showing these kinds of changes in general?
- Give the participant the debriefing form, ask him or her to sign the payment of subjects form thank him, pay him / her, shake his / her hand, and see him out.
- Write down any extra comments about that participant's performance.
- Prepare for the next participant.

A.2 Consent form

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

James Tam, Department of Computer Science, University of Calgary
Phone: (403) 220-3532, tamj@cpsc.ucalgary.ca

Lorin McCaffrey, Tec4 Systems Inc.
Phone: (403) 283-8876, Fax: (403) 283-1998, lorin@tec4.ca

If you have any questions not satisfactorily answered by the primary researchers concerning your participation in this project, you may contact the Office of the Vice-President (Research), University of Calgary, and ask for Pat Evans, (403) 220-3782.

Participant

Date

Investigator/Witness (optional)

Date

A copy of this consent form will be given to you to keep for your records if you request it. This research has the ethical approval of the Department of Computer Science and the University of Calgary.

A.3 Pretest questionnaire

1. Which of the following apply to you?

- I am in the SENG (Software Engineering) department but have no involvement in industry.
- I am in the SENG (Software Engineering) department and also use UML in industry.
- I am currently taking undergrad classes dealing with software engineering and/or UML (CPSC333 or CPSC451, for example).
- I work in a software engineering field and use UML as part of my job.
- Other: _____

2. How would you gauge your knowledge of software engineering and the Unified Modeling Language (UML) in relation to your peers?

I don't know UML	Novice	Average	Above Average	Expert
1	2	3	4	5

A.4 Post-Scenario Questions

Scenario #: _____

Mechanism: Icons

Replay

Storyboard

Change Index

1. Briefly explain to me how this change display mechanism works.

2. What items have changed since the last time you worked on the project? How have those items changed?

3. Can you tell which of your team member made the changes, and when they made those changes?

4. Keeping in mind the role that you are playing, how effective do you think the change indicators are in helping you keep track of the changes?

Useless	1	2	3	4	5	Very Effective
---------	---	---	---	---	---	----------------

5. Please describe why you gave the change indicator this rating.

6. How would you modify this change display mechanism to better keep track of changes?

7. Miscellaneous comments.

A.5 Post-Test Questions

1. Which of the four change display mechanisms did you prefer? Why was it better than the others?

2. If this was a project that you were really working on what other techniques would you use to keep up with the changes made to the project? This includes any non-electronic methods.

3. *Do you have any additional comments or questions about these change display mechanisms, or the task of showing these kinds of changes in general?*
