Expandable Grids: A user interface visualization technique and a policy semantics to support fast, accurate security and privacy policy authoring

Robert W. Reeder

July 2008
CMU-CS-08-143

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

This research was sponsored by the National Science Foundation under grant nos. CNS-0433540 and CNS-0627513, Army Research Office contract no. DAAD19-02-1-0389 and an IBM Open Collaborative Research grant.

Keywords: security; privacy; visualization; information visualization; policy; usability; P3P; policy authoring

This thesis is dedicated to

my parents,
from whom I have learned more than N years of graduate school could ever teach me,

and my sister,
who was always there for me when my Black Friday letters came.

Abstract

This thesis addresses the problem of designing user interfaces to support creating, editing, and viewing security and privacy policies. Policies are declarations of who may access what under which conditions. Creating, editing, and viewing—in a word, authoring—accurate policies is essential to keeping resources both available to those who are authorized to use them and secure from those who are not. User interfaces for policy authoring can greatly affect whether policies match their authors' intentions; a bad user interface can lead to policies with many errors, while a good user interface can ensure that a policy matches its author's intentions. Traditional methods of displaying security and privacy policies in user interfaces are deficient because they place an undue burden on policy authors to interpret nuanced rules or convoluted natural language.

We introduce the Expandable Grid, a novel technique for displaying policies in a user interface. An Expandable Grid is an interactive matrix visualization designed to address the problems that traditional policy-authoring interfaces have in conveying policies to users. This thesis describes the Expandable Grid concept, then presents three pieces of work centered on the concept:

- a design, implementation, and evaluation of a system using an Expandable Grid for setting file permissions in the Microsoft Windows XP operating system;
- a description and evaluation of a file-permissions policy semantics that complements the Expandable Grid particularly well for reducing policy-authoring errors; and
- a design, implementation, and evaluation of a system using an Expandable Grid for displaying website privacy policies to Web users.

The evaluations of the Expandable Grid system for setting file permissions and its associated policy semantics show that the Expandable Grid can greatly improve the speed and accuracy with which policy authors complete tasks compared to traditional policy-authoring interfaces. However, the evaluation of the Expandable Grid system for displaying website privacy policies suggest some limitations of the Grid concept. We conclude that the Expandable Grid is a beneficial promising approach to policy-authoring interface design, but that it must be applied with care and tailored to each domain to which it is applied.

Thesis Committee

Lorrie Faith Cranor (Chair)
Institute for Software Research
Engineering and Public Policy Department
Carnegie Mellon University

Jason I. Hong
Human-Computer Interaction Institute
Carnegie Mellon University

Michael K. Reiter
Computer Science Department
University of North Carolina at Chapel Hill

Daniel P. Siewiorek
Human-Computer Interaction Institute
Computer Science Department
Carnegie Mellon University

Clare-Marie Karat
IBM T.J. Watson Research Center

Acknowledgements

Although I appear as the sole author of this thesis, it is really a collective effort. I am indebted to many great people who have contributed to the work herein and supported me personally throughout my time in graduate school.

First I would like to thank my advisor, Lorrie Cranor. A good advisor makes all the difference in graduate school, and I found an exceptional one in Lorrie. She has contributed intellectually to just about every part of this thesis, including the graphical design of Expandable Grid interfaces, the experimental design of user studies, the ideas for our policy semantics, and the interpretation and presentation of our results. But her contributions have gone far beyond the mere content of the thesis. Lorrie has provided advice, encouragement, pressure, enthusiasm, and understanding at all the right times. When I needed resources, she provided them; when I needed direction, she knew which way to go. She is perhaps the most reasonable person I have ever worked with, and there is no one whose opinion I respect more than hers. I sincerely thank her for all she has done to make this thesis possible and for all that she has done for me personally and professionally. It is a great honor to be Lorrie's first PhD student.

All of my committee members have contributed to this work. Mike Reiter has provided excellent guidance on the file-permissions Expandable Grid work, particularly in the search for a policy semantics. Mike set us in the direction of teasing apart the effects of the Grid's presentation aspects from its underlying policy semantics, and of clearly defining the behavior of our semantics in the presence of group and folder dynamic changes. Mike's ability to mix intellectual heft with humility and humor has made working with him a great pleasure, and one of the most valuable parts of my experience at CMU. Jason Hong has diligently read drafts of my proposal, my papers, and this thesis and has always provided remarkably insightful feedback. His influence can be seen especially in the chapters on related work and the Expandable Grid concept. I also thank Jason for personal advice he provided on the job search and research careers. Clare-Marie Karat was my mentor during my internship the IBM T.J. Watson Research Center in the summer of 2006. She and John Karat helped me to a tremendously successful and enjoyable summer. It was at IBM that I first conceived of the Expandable Grid while working with the SPARCLE system. I also thank Clare-Marie and John for helping me see the path through some difficult issues during that summer, which ultimately turned my graduate school career around. Dan Siewiorek's feedback at my proposal led to our implementing sorting-by-color in the file-permissions Expandable Grid. I thank Dan for his service on my committee and appreciate his feedback on my work.

I have been very fortunate to work with the seven other students in the CUPS lab: Patrick Kelley, Kami Vaniea, Aleecia McDonald, Steve Sheng, Serge Egelman, Ponnurangam "PK" Kumaraguru, and Janice Tsai. Patrick, Kami, Aleecia, and Steve have made substantial contributions this thesis. Patrick designed and implemented the P3P Expandable Grid and provided excellent advice on the graphical design of the file-permissions Expandable Grid. He has been an enthusiastic collaborator and a good friend in the two years we have worked together. Kami implemented the very first working version of an Expandable Grid, and her technical aptitude saved the day when a problem with Windows XP taking minutes to resolve SIDs threatened to derail my last user study. I have valued our interactions as the two student members of the Grey usability team and I greatly value her friendship. Aleecia helped design and run the P3P Expandable Grid study. I have greatly benefited from our discussions about experimental design, statistical analysis, and life as a grad student. I thank Aleecia for sharing her humor and her outlook on life. Steve Sheng helped with statistics for the first file-permissions Expandable Grid study. Late nights in the CIC building were always more fun when Steve was around to share them. Serge has been a good friend and a fun person to know. He has provided much relief from the rigors of graduate school, but has still forged his own impressive intellectual path. I have appreciated and learned from PK's dedication to his work and insistence on excellence in research methods and presentation. Janice has been a good friend who impresses me with her tenacity in her work.

I owe a great deal of thanks to Lujo Bauer, with whom I have worked closely on the Grey project. Lujo has made substantial contributions to the work on the file-permissions Expandable Grid and the policy semantics for the Grid. He has also given me detailed feedback on my papers and on this thesis. I have learned much from Lujo's dedication to his work, precision in his writing, and generous nature, and I have greatly enjoyed his sense of humor. Due to Lujo's influence, I use far more hyphens in my writing, which is a good thing. We all know the world can use more hyphens.

A number of undergraduate research assistants have helped me complete the work for this thesis, and I acknowledge and heartily thank all of them for their contributions. Heather Strong, Keisha How, Kelli Bacon, and Chris Jackson served as RAs on the Expandable Grid project in the summer of 2007. Heather is almost solely responsible for the implementation of the Grey Expandable Grid. Keisha contributed all kinds of help, including scripting, Web programming, and configuration for the first file-permissions Grid study and the P3P Grid study. Keisha also wrote the text for the large-scale tasks in the file-permission Grid study, including my favorite, the Rita task, in which a nefarious member of the Music Department has replaced the snare drum part for Carmen with a fake part that sounds like Jingle Bells. Kelli contributed to the code of the file-permissions Expandable Grid and ran her own user study based on the Grid. Chris Jackson is largely responsible for the implementation of an Expandable Grid for PeopleFinder. I worked with Danielle Chang, Tony Poor, Ilkyoo Choi, and Jerry Feng during the 2007-2008 academic year. Danielle and Tony helped code the results of the P3P Grid lab study. Danielle, Ilkyoo, and Jerry worked on a design to overcome the drawbacks of vertical labels in the file-permissions Grid. Tony also helped design the new dog-ear features of the file-permissions Grid that we added with our new policy semantics. I worked with Jerry, Joanna Bresee, and Daniel Rhim in the summer of 2008. Joanna designed the new Expandable Grid legend, helped with the design of the dog-ears, wrote some of the tasks for the second policy semantics study, and ran the first few participants through the study, all in a few short weeks. Jerry and Daniel ran the remainder of the study, recorded the results, and coded errors. All of the undergraduate RAs I have worked with have exceeded my expectations and showed remarkable competence, creativity, and enthusiasm; this thesis is full of their contributions.

Brandon Salmon has been a great friend and collaborator. Brandon designed Perspective, a distributed semantic file system, and applied the Expandable Grid concept to the design of a user interface for setting storage policies in Perspective. I have been delighted to see the Grid concept applied successfully to a domain I had not even imagined applying it to. Conversations with Brandon and Eric Toan have influenced my thinking about the design and implementation of the Grid.

I have already acknowledged Clare-Marie and John Karat for their guidance during my summer at IBM. I would also like to thank Peter Malkin, my co-inventor on a patent application on the Expandable Grid, and Carolyn Brodie, both of whom helped make it a great summer.

As any good graduate student should, I thank the excellent staff in the CS Department for all of their help over my years at CMU. Sharon Burks deserves special mention for some personal counsel that helped turn my graduate school career around. I also thank the ever-vibrant Catherine Copetas and Deb Cavlovich for all their hard work to make CMU's CSD a great place to be.

I thank my many friends at CMU who helped make my years here so enjoyable. Cynthia Kuo read my papers and provided much needed encouragement and sympathy throughout my graduate school career. Melissa Lawrie has been a patient friend as I've written this thesis; she has also kindly proofread drafts. Andrew Faulring has been a loyal friend, and I have often made use of his good research sense. Charlie Garrod, Adam Fass, and Adam Wierman have been sources of inspiration and fun. I especially thank Adam Wierman for providing the LaTeX template for this document. My "inner circle" of friends, Rachel Gougian, Julia McGinn Rodriguez, Christine Taylor, Jen Clarke, and Rob Simmons have made the last two years terrific fun.

Finally, I would like to thank my family, to whom this thesis is dedicated. My parents have always provided love and support through the vicissitudes of graduate school and life, and my sister is the best sister in the world.

Thank you all.

Table of Contents

List of Figures

List of Tables

PART I

Motivation and Background

CHAPTER 1

Introduction

Policies are fundamental to providing security and privacy in a wide variety of applications including file systems, Web browsing, file-sharing networks, firewalls, online social networks, online health databases, and location sharing. Such applications provide control over security and privacy through policy-based systems that take requests for resources and grant or deny access to those resources according to a policy. Since policy-based systems govern the tradeoff between system functionality and data security or privacy, policies must be accurate in order to ensure that this tradeoff is made as desired. Inaccurate policies can lead to denial of service on the one hand and to compromised resources on the other. Since humans author these policies, it is important to provide usable interfaces that allow human users to create accurate and comprehensive policies. However, research and everyday experience have demonstrated that the user interfaces we have today for policy authoring are not usable.

At the same time that many have acknowledged the inadequacy of today's policy-authoring interfaces, the need for usable policy-authoring interfaces is on the rise. As computing becomes increasingly collaborative, distributed, and pervasive, data becomes more available, and authoring and understanding the policies that control access to this data becomes both more necessary and more challenging. While making data available enables many beneficial applications, the very availability of data creates opportunities for abuse. Policies must be in place to prevent parties with ill intentions from accessing data and other computing resources. However, as more applications require policies to be in place, there are fewer experts, such as system administrators, relative to demand to author these policies. As a result, policy authoring increasingly falls to end users or non-technical people, groups of users who are novice or occasional policy authors. While dedicated system administrators might be able to invest the time to learn and use complex user interfaces, this rising class of novice and occasional policy authors cannot be expected to do the same. User interface designers must find designs that enable these policy authors to create their intended policies without investing significant time and building significant skills.

Despite the need for usable policy-authoring interfaces, there is evidence that today's widely-used policy-authoring interfaces are prone to serious errors. The "Memogate" scandal, in which staffers from one political party on the United States Senate Judiciary Committee stole the opposing party's confidential memos from a file server that the two parties shared, was caused in part by an inexperienced system administrator's error using the Windows NT interface for setting file permissions [132]. A study we conducted showed cases in which users of the Windows XP file permissions interface made errors that exposed files to unauthorized access [85]. Good and Krekelberg showed that users unwittingly shared confidential personal files due to usability problems with the KaZaA peer-to-peer file-sharing application's interface for specifying shared files [55]. Herzog and Shahmehri revealed numerous usability problems with policytool, a policy-authoring interface for setting permissions for Java programs [57]. In the domain of website privacy policies, a number of studies have shown that natural language interfaces are a poor means for conveying privacy policies to website visitors [7, 64, 100, 91].

1.1 Problem: Usability of policy-authoring interfaces

This thesis seeks to improve upon two of today's dominant paradigms for presenting policies to policy authors. First is what we call the list-of-rules paradigm, which is widely used in interfaces for authoring policies in domains such as file access control, firewall, enterprise privacy, and location disclosure policies. Second is natural language, particularly as it is used to present privacy policies to website visitors.

The main problem with list-of-rules policy-authoring user interfaces is that they force policy authors to focus on individual rules, which are only a portion of a policy, rather than being centered around the policy as a whole. A policy can be thought of as a set of rules that dictate what requests for resources will be granted in a computing system. Authoring a policy is a matter of specifying this set of rules, so it is natural to center an interface around a list of rules. A list-of-rules interface allows an author to select one rule to create or edit and presents the author with information about that rule. However, list-of-rules interfaces hide contextual information from authors as they are authoring a specific rule, even though users often need such contextual information to author rules correctly. For example, rules may conflict with one another, so authors need information about other rules to understand how a given rule will be affected by others. As another example, policies usually have a default rule, i.e., a rule that applies to any requests not explicitly covered by written rules. Users need to understand default rules in order to know what cases they need to write rules for. In past work, I have shown both that a lack of needed information is responsible for user errors in policy authoring and that providing this information in an easily accessible form can prevent many errors [85].

There are also several potential usability problems with natural language policy presentations. Although presenting privacy policies to consumers in a comprehensible manner is important from both consumer protection [45] and commercial [27, 91, 131] perspectives, privacy policies are usually presented in legalistic, convoluted language [100], and are often written at a college or higher reading level [7, 64]. To make matters worse, presentations vary from website to website; thus, from the Web user's point of view, every privacy policy is as hard to read as the last one, and it is very difficult to compare privacy policies across competing websites.

1.2 Proposed solution: the Expandable Grid

To address some of the deficiencies of today's policy-authoring interfaces, we propose a new design paradigm: the Expandable Grid. The Expandable Grid is an interactive matrix visualization of a policy. A implementation of the Expandable Grid concept in an interface for setting file permissions is shown in Figure 1.1. The motivation behind the Expandable Grid concept is to center policy-authoring interfaces around a standardized, graphical visualization of the entire effective policy. The visualization-centered approach we advocate applies many of the principles of information visualization to provide quick access to the decision a policy would make for any access request—not just the immediate rule that a user is editing. Providing this quick access allows the interface to show all the contextual information relevant to a given rule. By showing *effective* policy, our approach does the work of combining rules together, thus relieving policy authors of the burden of figuring out the results of rule conflicts (which occur when multiple rules apply to an access request) and default rules (which take effect when no rule applies to an access request) in their heads. Using a standardized graphical layout also promises to prevent problems associated with natural language such as convoluted language and to allow for different websites' policies to be compared easily.

1.3 Applications of the Expandable Grid concept

The Expandable Grid concept could conceivably be applied to a wide variety of policy domains, but this thesis focuses on two: file permissions in the Windows XP operating system and website privacy policies. These two domains were chosen because their respective policy-authoring interfaces are widely used and because they are very different from each other. In this latter respect, they provide a test of the generalizability of the Expandable Grid concept. We designed and implemented a user interface based on the Expandable Grid concept for each of these applications. We conducted user studies to evaluate the interfaces over a broad range of policy-authoring tasks. We discuss each of the two applications and their respective Expandable-Grid-based interfaces below.

1.3.1 An Expandable Grid for file permissions

In a basic file access-control system, there are resources, principals, and actions. Resources include files and folders, the data to which access is to be controlled. Principals are system users and groups of system users. Actions are operations that may be performed on resources, such as "read," "write," "execute," and "delete." A file permissions policy is a set of rules that state what principals may perform what actions on which resources. Rules may allow or deny access. For example, a rule may state that the user "jsmith" is allowed to read the file "budget.xls," or that the user "mary" is denied access to write to the file "privateData.txt." Since a file system may have many principals, file access control systems typically provide a means for constructing *groups*, which are collections of users. For instance, users might be placed into groups or roles according to their position in an organization, such as "Managers," and "Employees." Similarly, files may be grouped into hierarchical folders. Groups and folders allow policy authors to write rules that apply to all components of a composite value, such as a rule that denies delete capability to all employees for files in a folder called "Critical files."

The Windows XP file permissions interface provides a good example of a list-of-rules policy-authoring interface and its limitations. To describe the limitations of the list-of-rules approach, it is convenient to start with an example task. Here, we describe the Jana task, one of the 20 tasks we designed for the user study described in Chapter 6. Our statement of the Jana task reads as follows:

> Jana, a Theory 101 TA, complained that when she tried to change the Four-part Harmony handout to update the assignment, she was denied access.
>
> Set permissions so that Jana can read and write the *Four-part Harmony.doc* file in the *Theory 101\Handouts* folder.

Figure 1.2 shows the state of the Windows XP file permissions interface after a common error that policy authors make. Authors create a rule stating that Jana is allowed read and write access to the *Four-part*

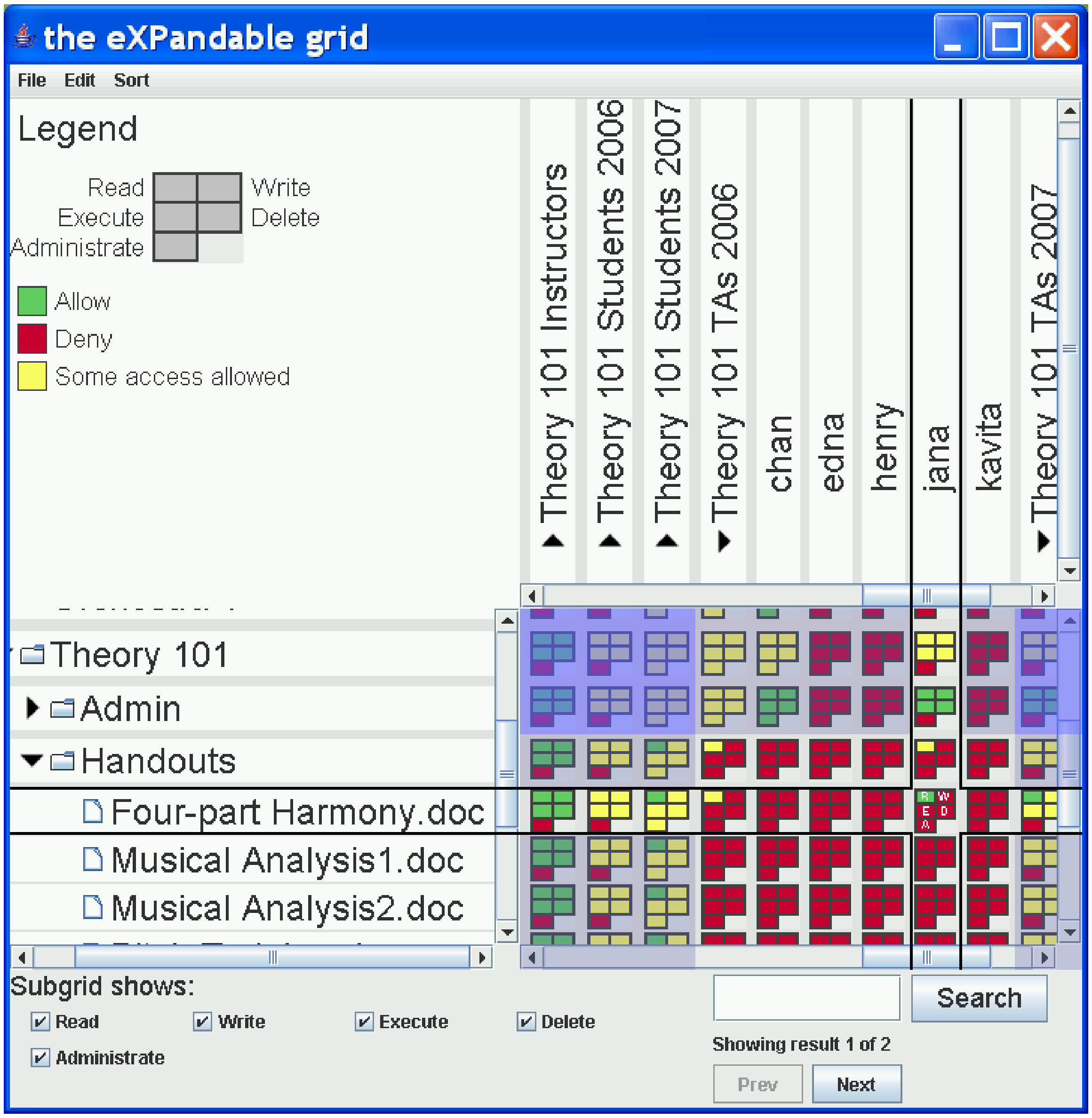
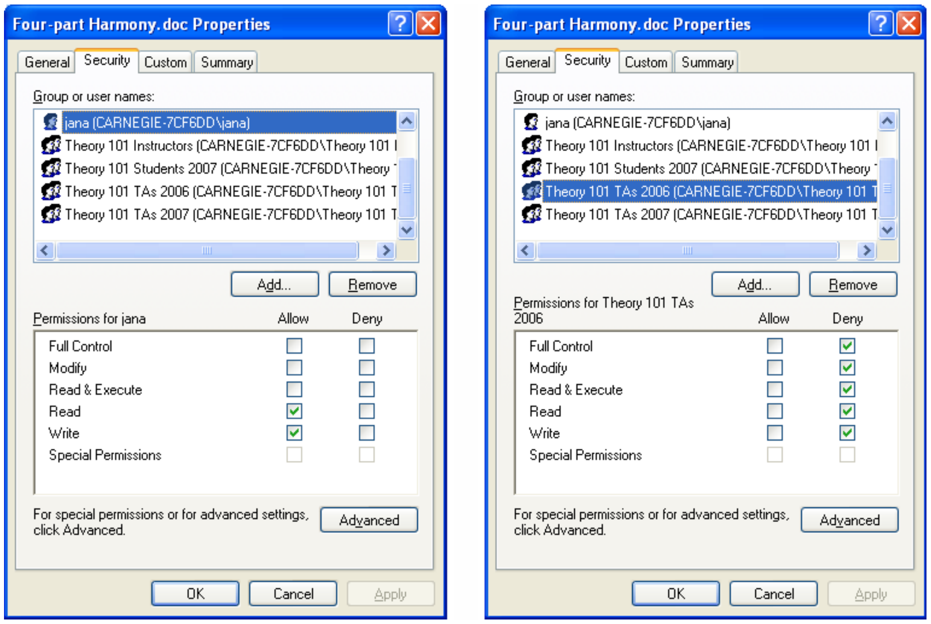


Figure 1.1: *Our Expandable Grid interface for setting file permissions in Windows XP. The interface shows principals along the upper axis, resources along the left-hand axis, and the effective policy applying to the principals and resources in the colored squares in the grid itself.*



Four-part Harmony.doc Properties

General | Security | Custom | Summary

Group or user names:

- jana (CARNEGIE-7CF6DD\jana)
- Theory 101 Instructors (CARNEGIE-7CF6DD\Theory 101 I
- Theory 101 Students 2007 (CARNEGIE-7CF6DD\Theory
- Theory 101 TAs 2006 (CARNEGIE-7CF6DD\Theory 101 T
- Theory 101 TAs 2007 (CARNEGIE-7CF6DD\Theory 101 T

Add... | Remove

Permissions for jana	Allow	Deny
Full Control	☐	☐
Modify	☐	☐
Read & Execute	☐	☐
Read	✓	☐
Write	✓	☐
Special Permissions	☐	☐

For special permissions or for advanced settings, click Advanced. Advanced

OK | Cancel | Apply

Four-part Harmony.doc Properties

General | Security | Custom | Summary

Group or user names:

- jana (CARNEGIE-7CF6DD\jana)
- Theory 101 Instructors (CARNEGIE-7CF6DD\Theory 101 I
- Theory 101 Students 2007 (CARNEGIE-7CF6DD\Theory
- Theory 101 TAs 2006 (CARNEGIE-7CF6DD\Theory 101 T
- Theory 101 TAs 2007 (CARNEGIE-7CF6DD\Theory 101 T

Add... | Remove

Permissions for Theory 101 TAs 2006	Allow	Deny
Full Control	☐	✓
Modify	☐	✓
Read & Execute	☐	✓
Read	☐	✓
Write	☐	✓
Special Permissions	☐	☐

For special permissions or for advanced settings, click Advanced. Advanced

OK | Cancel | Apply

Figure 1.2: *The Windows XP file permissions interface, an example of a list-of-rules policy-authoring interface. These screenshots give an example of why it is difficult for users to understand an effective policy in the presence of a rule conflict.*

Harmony.doc file. Looking at the interface as pictured in the left-hand screenshot in Figure 1.2, it appears Jana is now allowed to read and write the file. What policy authors miss, however, is that Jana is a member of the group Theory 101 TAs 2006; there is a rule, as can be seen in the right-hand screenshot in Figure 1.2 that denies this group access to read and write the file. Thus, two rules apply to Jana, one allowing her access and one denying her access; this situation is a rule conflict. In Windows, rules that deny access take precedence in conflicts with rules that allow access, so in Jana's case, she is denied access to the file. To understand Jana's situation, the policy author must understand three things: first, that Jana is in the group Theory 101 TAs 2006; second, that Theory 101 TAs 2006 are denied write access to the file; and third, that Deny permissions take precedence over allow permissions. Once these three items are understood, they need to be remembered and applied to determine the correct method for completing this task: removing the deny rule for Theory 101 TAs 2006 so that the rule allowing Jana access will become operative, while the other members of the group will still be denied access to the file by default. However, the Windows list-of-rules interface makes it difficult for the policy author to obtain these three pieces of information. Group membership information is only available in the Computer Management application, which is wholly separate from the file permissions interface, and thus rarely found; the rule applying to Theory 101 TAs 2006 is easily overlooked; and the deny-takes-precedence rule essentially has to be inferred. It is difficult for novice or occasional policy authors to complete this task correctly, even though it is a fairly typical scenario.

The XP file-permissions interface has the following drawbacks:

- It only shows the policy for one resource at a time;

- It shows rules, not how the rules combine to form the actual access that will be allowed to users;
- It can only show one rule at a time;
- It gives no indication of what happens by default when no rule applies to a particular user;
- It gives no indication of when rules conflict or how conflicts are resolved;
- It does not show group membership information.

The XP file-permissions interface is perhaps the most widely distributed list-of-rules interface and it is the one on which we concentrate in this thesis, but it is by no means the only representative of the list-of-rules paradigm. The paradigm has also been adopted by a Linux open source file-permissions interface [60], the file-permissions interface in the Mac OS X Server [9], a research interface for location disclosure policies [76], and interfaces for setting firewall policies [6].

A screenshot of our Expandable Grid interface for setting file permissions can be seen in Figure 1.1. The tree along the vertical axis at the left of the interface shows the resources in a file system. The rotated tree along the horizontal axis at the top of the interface shows the principals. At the intersection of these two trees is a grid that shows the access each principal has to each resource. Grid cells each correspond to one principal and one resource. Each grid cell is further subdivided into a "subgrid," a large square divided into smaller boxes. The subgrids allow the interface to show a third policy dimension, as long as that dimension has only a handful of values. In our interface, the subgrid represents five different types of access. The upper left box in each group of five boxes indicates read access, the upper right box indicates write access, the middle left box indicates execute access, the middle right box indicates delete access, and the lower box indicates administrate access. Green boxes (which appear as a medium grey in greyscale) indicate access that is allowed, red boxes (which appear dark grey in greyscale) indicate access that is denied, and yellow boxes (which appear light grey in greyscale) indicate that items lower in one or both trees have a mixture of allowed and denied access.

The screenshot in Figure 1.1 shows an intermediate state the interface would be in when the Jana task has been half-completed. The grid cell at the intersection of "jana" and the "Four-part Harmony.doc" file is highlighted in the crosshairs. In the highlighted grid cell, the upper-left box is green, indicating that Jana has already been given read access to the "Four-part Harmony.doc" file, while the upper-right box is red, indicating that Jana does not yet have write access. Clicking on the upper-right box will give Jana write access and cause the box to turn green. In contrast to the checkboxes in the Windows file permissions interface, the colored boxes of the grid indicate the actual access a user will have to a resource, after all policy settings have been taken into account. When a red box is clicked on, the policy is changed so that Jana is now allowed access, and the box turns green; there is no need to be aware of other rules that may interfere with the intention to allow her access. Thus, it is much easier to complete the Jana task using the Expandable Grid interface and to be certain that it has been completed correctly.

The Expandable Grid interface addresses the drawbacks of the list-of-rules approach:

- It shows the whole policy, for all resources and all principals, in one view;
- It shows the effective policy, which consists of the actual access decisions that will be made;
- By showing effective policy, it shows the combined effect of all rules;

- By showing effective policy, it shows how rule conflicts are resolved;
- It shows group membership information in the tree of principals.

We conducted a user study comparing our Expandable Grid interface with the Windows XP file permissions interface for a broad range of policy-authoring tasks and found the Grid interface vastly outperformed the Windows interface in speed and accuracy across most of the tasks. For example, in one task, the Grid achieved a 100% accuracy rate compared to 6% for Windows; in another task, the Grid achieved a 70% reduction in the time required to complete the task. The Grid interface for file permissions and the user study results are covered in detail in Chapter 6.

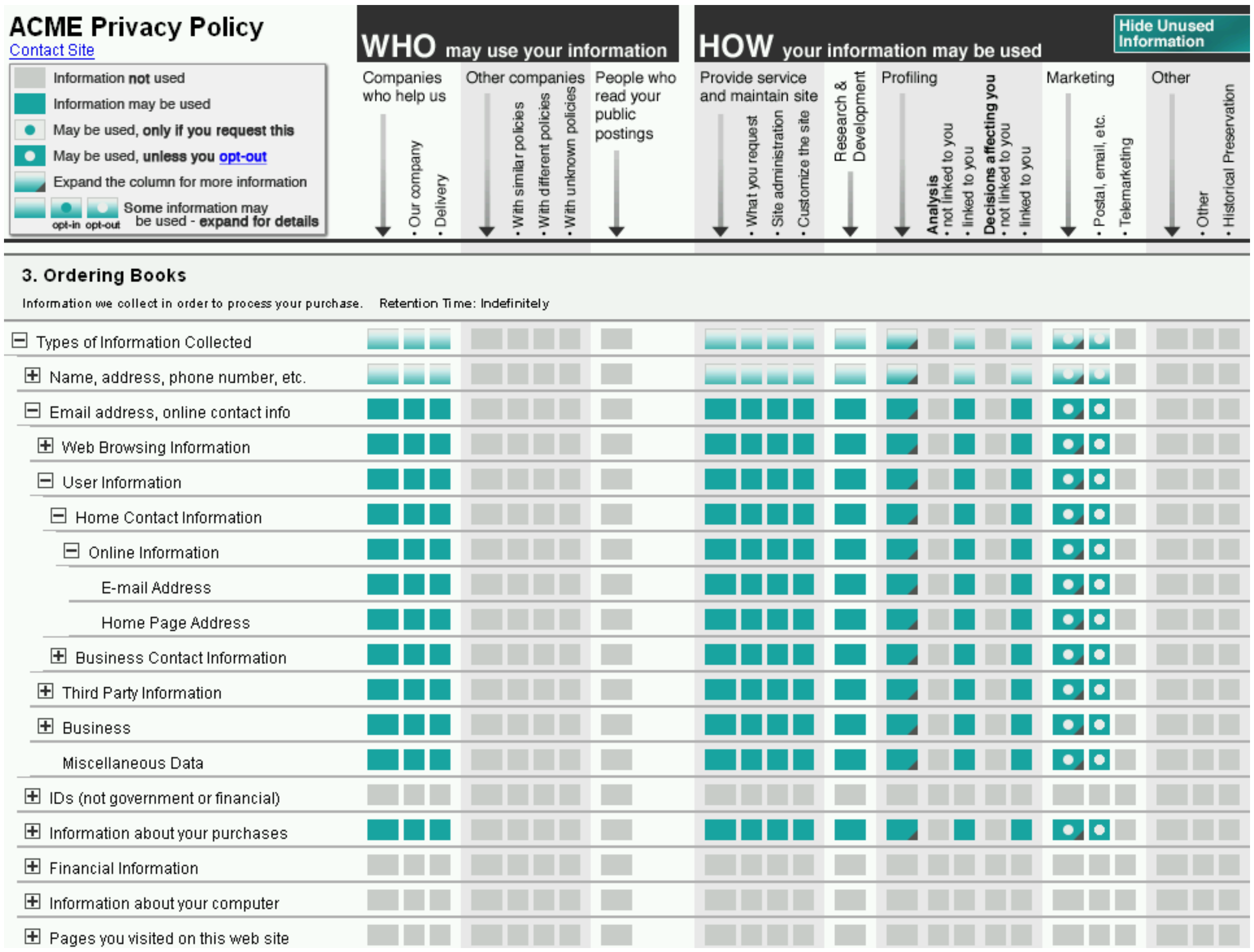
1.3.1.1 Policy semantics in the Expandable Grid

One of the key aspects to the Expandable Grid for file permissions is showing the effective policy. In the Jana task, Jana is initially in two groups: the group Theory 101 TAs 2007, which is allowed read and write access to the “Four-part Harmony.doc” file, and the group Theory 101 TAs 2006, which is denied that access. Thus, Jana has conflicting rules on the file. According to the policy semantics used by Windows, when rules conflict, the rule denying access takes precedence. The Grid allows policy authors to see that although Jana has one rule allowing her read and write access to the file and another rule denying her that access, she is actually denied access, as indicated by the red squares.

The Grid maintains the focus on the effective policy when it comes to changing the policy. Policy authors can click on a red square and turn it green. This operation was so simple that 100% of participants using the Grid in our user study successfully completed the Jana task. However, enabling the Grid to simply turn Jana’s square green required a small trick—we had to change the policy semantics so that when an author clicks the red square corresponding to Jana’s access to the “Four-part Harmony.doc” file, it sets a rule allowing access, and that rule takes precedence over the existing rule denying her access. We changed the semantics so that the most recently set rule takes precedence.

The success of the Expandable-Grid-based interface for setting file permissions in Windows XP led to the question of whether this success was due primarily to the Grid as a presentation technique or to the Grid’s new semantics. In a followup study, described in Chapter 7 we show that the answer is both: the Grid with Windows semantics performs better than the native Windows interface, and the Grid with the new semantics performs even better.

Given that our altered semantics did in fact lead to a usability gain, we asked what semantics was best from a policy-authoring usability perspective. Our recency semantics appears to have some drawbacks, especially when group memberships change and files can be moved from one folder to another. In Chapter 7, we propose a *specificity semantics* in which the most specific rule—i.e., the rule applying to a specific user instead of a group, or a specific file or subfolder instead of a higher-level folder—takes precedence in a rule conflict. We implemented this semantics in our Expandable Grid, and in Chapter 7 we describe the semantics and a user study comparing the Grid interface with the specificity semantics to the Grid interface with XP semantics and to the Windows XP file permissions interface. Although there has been prior work on using specificity to resolve conflicts [54], we believe we are the first to evaluate a semantics with an actual user study.



ACME Privacy Policy
Contact Site

▢	Information **not** used
■	Information may be used
●	May be used, **only if you request this**
◉	May be used, **unless you opt-out**
▨	Expand the column for more information
◪ ● ◉ opt-in opt-out	Some information may be used - **expand for details**

WHO may use your information | **HOW** your information may be used | Hide Unused Information

	Companies who help us	• Our company	• Delivery	Other companies	• With similar policies	• With different policies	• With unknown policies	People who read your public postings	Provide service and maintain site	• What you request	• Site administration	• Customize the site	Research & Development	Profiling	**Analysis** • not linked to you	• linked to you	**Decisions affecting you** • not linked to you	• linked to you	Marketing	• Postal, email, etc.	• Telemarketing	Other	• Other	• Historical Preservation
3. Ordering Books Information we collect in order to process your purchase. Retention Time: Indefinitely																								
⊟ Types of Information Collected	▨	▨	▨	▢	▢	▢	▢	▢	▨	▨	▨	▨	▨	◪	▢	▨	▢	▨	◉◪	◉	▢	▢	▢	▢
⊞ Name, address, phone number, etc.	▨	▨	▨	▢	▢	▢	▢	▢	▨	▨	▨	▨	▨	◪	▢	▨	▢	▨	◉◪	◉	▢	▢	▢	▢
⊟ Email address, online contact info	■	■	■	▢	▢	▢	▢	▢	■	■	■	■	■	◪	▢	■	▢	■	◉◪	◉	▢	▢	▢	▢
⊞ Web Browsing Information	■	■	■	▢	▢	▢	▢	▢	■	■	■	■	■	◪	▢	■	▢	■	◉◪	◉	▢	▢	▢	▢
⊟ User Information	■	■	■	▢	▢	▢	▢	▢	■	■	■	■	■	◪	▢	■	▢	■	◉◪	◉	▢	▢	▢	▢
⊟ Home Contact Information	■	■	■	▢	▢	▢	▢	▢	■	■	■	■	■	◪	▢	■	▢	■	◉◪	◉	▢	▢	▢	▢
⊟ Online Information	■	■	■	▢	▢	▢	▢	▢	■	■	■	■	■	◪	▢	■	▢	■	◉◪	◉	▢	▢	▢	▢
E-mail Address	■	■	■	▢	▢	▢	▢	▢	■	■	■	■	■	◪	▢	■	▢	■	◉◪	◉	▢	▢	▢	▢
Home Page Address	■	■	■	▢	▢	▢	▢	▢	■	■	■	■	■	◪	▢	■	▢	■	◉◪	◉	▢	▢	▢	▢
⊞ Business Contact Information	■	■	■	▢	▢	▢	▢	▢	■	■	■	■	■	◪	▢	■	▢	■	◉◪	◉	▢	▢	▢	▢
⊞ Third Party Information	■	■	■	▢	▢	▢	▢	▢	■	■	■	■	■	◪	▢	■	▢	■	◉◪	◉	▢	▢	▢	▢
⊞ Business	■	■	■	▢	▢	▢	▢	▢	■	■	■	■	■	◪	▢	■	▢	■	◉◪	◉	▢	▢	▢	▢
Miscellaneous Data	■	■	■	▢	▢	▢	▢	▢	■	■	■	■	■	◪	▢	■	▢	■	◉◪	◉	▢	▢	▢	▢
⊞ IDs (not government or financial)	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢
⊞ Information about your purchases	■	■	■	▢	▢	▢	▢	▢	■	■	■	■	■	◪	▢	■	▢	■	◉◪	◉	▢	▢	▢	▢
⊞ Financial Information	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢
⊞ Information about your computer	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢
⊞ Pages you visited on this web site	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢

Figure 1.3: *A screenshot of our P3P Expandable Grid for displaying website P3P privacy policies.*

1.3.2 An Expandable Grid for P3P

The Platform for Privacy Preferences (P3P) is a formal, machine-readable language for expressing privacy policies in a manner that is standardized across websites. P3P makes it possible to display policies in a standardized way. We have developed an Expandable-Grid-based interface for presenting website P3P privacy policies to Web end users.

P3P defines a set of data practices in which an organization might engage, and a P3P policy specifies which of those practices a specific organization engages in. For example, one potential data practice is collecting a consumer's email address and sharing it with other companies to contact the consumer for marketing purposes. Each potential data practice defined by P3P consists of three primary attributes: data category, recipient, and purpose. So, in the example data practice, "online contact information" is the data category, "other companies" is the recipient, and "contact for marketing" is the purpose.

Our P3P Expandable Grid can be seen in Figure 1.3. Since P3P data practices are defined by the three primary data-specific assertions, we used the P3P data categories, purposes, and recipients as the labels along the axes of our P3P Expandable Grid. Colored squares in the grid cells indicate those data practices in which the organization issuing the represented policy engages; teal (dark grey in greyscale) squares indicate practices in which the organization does engage, and light grey squares indicate practices in which the organization does not engage. Teal squares with a white dot indicate data practices in which the organization engages unless a website users opts out of the practice.

We conducted a user study over the Web comparing the P3P Expandable Grid with natural language as a means for presenting privacy policies to Web users. The study showed that participants performed no better on a set of comprehension questions using the P3P Expandable Grid presentation than they did using the natural language presentation. This result was somewhat surprising, given the Grid success in a file-permissions interface. In a follow-up lab study, we identified several specific reasons our implementation of the Expandable Grid for P3P did not work well. The studies and their results are discussed in Chapter 8.

1.4 Thesis statement

The objective of this thesis is to test the hypothesis:

> **Policy-authoring user interfaces using techniques centered on visualization of a full, effective policy lead to faster, more accurate performance on policy-authoring and policy-viewing tasks than do today's list-of-rules and natural-language policy-authoring interfaces.**

Results of our user studies partially support the hypothesis. Specifically, our Expandable-Grid-based user interface for setting file permissions vastly outperformed a list-of-rules-based interface, but our Expandable-Grid-based user interface for presenting website privacy policies did not outperform a natural-language-based presentation. We discuss reasons the Grid worked well in one domain but not the other and draw conclusions about how and when to apply the Expandable Grid concept.

1.5 Outline of the thesis

This chapter has introduced some of the problems around policy-authoring interface design and our solution, the Expandable Grid. Chapter 2 reviews related work in policy-authoring interface design, usable security, information visualization, and policy languages. Chapters 3 and 4 present some of our preliminary work that led to the Expandable Grid concept and the other ideas in this thesis. Chapter 5 presents the Expandable Grid concept, lists characteristics of various policy domains, and discusses how these characteristics make some policies suited and others not suited to being displayed in an Expandable Grid. Chapter 6 describes the file-permissions Expandable Grid interface in detail and presents the results of the user study we conducted to evaluate the interface. Chapter 7 describes the file permissions policy semantics we developed to best take advantage of the Grid's properties and presents results of user studies we conducted to evaluate the effect of the semantics on the usability of the Grid. Chapter 8 describes the P3P Expandable Grid in detail and presents results of a Web-based study we conducted to compare the P3P Expandable Grid policy presentation with a natural-language-based policy presentation. Chapter 9 reviews some additional applications of the Expandable Grid beyond file permissions and P3P. Chapter 10 summarizes results of the individual studies, discusses what these results say about when and how the Expandable Grid concept can best be applied, and mentions future directions for work on applying the Expandable Grid and creating usable policy-authoring interfaces in general.

CHAPTER 2

Background and Related Work

This thesis is about using the Expandable Grid concept to build user interfaces to help policy authors write policies accurately and efficiently. Here, we discuss related work, which falls into four broad areas:

1. *Policy-authoring interface design.* This area includes the work most directly related to this thesis and includes approaches to developing usable interfaces in a variety of specific policy-authoring domains.
2. *HCISEC (human-computer interaction for security).* This area includes work generally related to usability issues around computer security applications. We discuss how our work on policy-authoring user interfaces fits into the broad themes of HCISEC.
3. *Cognitive science.* Work in the study of human cognition provides a theoretical basis to explain why our approach of presenting a visualization of the full, effective policy might help policy authors with their tasks.
4. *Information visualization.* Information visualization is a sub-discipline of human-computer interaction that has produced a variety of interactive techniques for displaying large amounts of data. Some of our work in policy visualization has borrowed from the principles and techniques of information visualization.

2.1 Policy-authoring interface design

Past work in designing user interfaces for policy-authoring applications has focused on specific domains. While researchers have produced some successful designs, prior work has not produced general techniques applicable to a variety of policy-authoring domains. Tabular visualization approaches similar to the Expandable Grid have been included as part of at least two systems for enterprise security and privacy policy authoring (SPARCLE and HP Select Access, described below), but they have not been evaluated in user studies nor generalized to policy-authoring domains aside from enterprise policy authoring.

The HP Select Access Policy Builder is an application for authoring enterprise security and privacy policies [94]. It includes a user interface that provides a visualization of a policy in a matrix indexed by principals on one axis and resources on the other. This is very similar to the idea of the Expandable Grid. However, the HP Policy Builder is limited to showing principals and resources along the axes; there is apparently no provision for showing any other policy attributes, such as actions (one of the common attributes in file access control) or purposes (one of the common attributes in privacy policies). Moreover, no user study evaluating the HP Policy Builder has been published.

Karat et al.'s SPARCLE system is concerned with the problem of enterprise privacy policy authoring [69, 68]. SPARCLE provides a natural-language input mechanism for authors to write privacy policy rules in English, and also provides a mechanism for authors to select rule elements from structured lists. SPARCLE provides a matrix-based full-policy visualization. SPARCLE's matrix visualization is limited, however, to displaying policies with no more than about five element values in each dimension without scrolling. The present work on the Expandable Grid concept was conceived as a means to make the SPARCLE policy visualization scale to much larger policies.

Zurko et al. designed the Adage system, a policy-based access control system for distributed computing systems [129, 153]. The Adage designers put a premium on usability, so Adage included the Visual Policy Builder (VPB), a graphical user interface for supporting policy authoring. Adage and the VPB supported operations fundamental to policy authoring: viewing and changing policies, viewing group memberships, and detecting and resolving conflicts. In its presentation of policies to users, however, Adage was a list-of-rules interface. Users could view single rules or lists of rules, but not the full policy implied by the rules. Zurko's usability testing revealed the need for a policy overview, and she suggested that research in "dynamic visualization" could be applied to policy visualization [152]. A visualization-centered interface was neither implemented nor tested in Adage, however.

Cao and Iverson presented Intentional Access Management (IAM) systems for access control [29]. They define IAM systems as systems that take a user's intention for a specific policy decision, convert that intention into possible rules that would result in the decision, and present the possible rules to the user. Thus, the user is freed from having to figure out when a rule conflict will occur. Cao and Iverson demonstrate that an implementation of IAM for WebDAV, a Web-based distributed file system, is an advance over the standard WebDAV Access Control List editor. However, Cao and Iverson's interface remains rule-centered; it does not provide a means for visualizing the full policy.

Balfanz developed the ESCAPE Web server, which provided an easy-to-use access control system for content published on the Web [11]. His system allowed a person to put content on the Web and send email announcements to people who were allowed to view the content. The ESCAPE server would automatically give access to the recipients of the emails. The ESCAPE server provided a user interface for viewing directories of Web content and the people who had access to those directories. The ESCAPE server, while not evaluated, would probably make Web access control policy authoring easier. However, its usability gains were due to simplifications of the access control model, namely, having only two input dimensions—users and folders—and not allowing for user grouping. These simplifications were probably appropriate for end-user Web publishing, but would not be appropriate for all policy-authoring applications.

The IBM P3P Policy Editor is typical of a list-of-rules policy authoring interface [40]. It presents Platform for Privacy Preferences (P3P) policies as lists of textual rules. While it is an excellent tool for experienced P3P policy authors, its design does not lend itself to finding a particular decision or verifying a property of the policy. To do so, one would have to read through the list of rules and assemble the policy in one's head.

Cranor et al. developed an interface for allowing people to specify what privacy practices they prefer in websites with which they interact [41]. Their approach to making their interface usable was to limit the choices available to a few that had been shown to be most important to people, such as how their health and medical information could be used and whether they would allow websites to contact them by telephone. Their approach was very successful, but is very specific to the Web-based privacy preferences domain.

Lederer et al. developed user interface prototypes for setting location-disclosure policies in a pervasive computing environment [76]. Their first prototype was rule-centered, but a study showed the prototype to be confusing. A second prototype, using a design drawn from instant messaging clients, listed every friend who had access to a user's location information. This latter prototype allowed users to view policies in a friend-centered way, but not by any other attribute, such as situation (users may want to reveal their location when they are at work, but not when they are out partying, for example). This limitation was probably acceptable for location-disclosure applications, as Lederer et al. found [77], but, in general, policy authors may want to view policies according to more than one attribute.

The Role Control Center (RCC) is an application for managing role-based access control policies [23, 48]. It includes a graphical user interface that shows the assignment of users to roles in a directed graph and allows for viewing policy either by principal or by resource. The assignment of users to roles is equivalent to what we have referred to as group memberships, so RCC supports viewing group memberships very well. RCC's ability to show policy by principal or by resource is an improvement over access-control interfaces, like the Windows XP file-permissions interface, that only allow for viewing policy by resource, but it still shows policy only in lists of rules. It does not have a single full-policy visualization.

In the domain of firewall policy-authoring interfaces, Mayer et al. developed Fang, a firewall analysis tool, and Bartal et al. (including some of the Fang authors) developed Firmato, a firewall management toolkit [86, 14]. Fang includes a user interface for querying the effective policy of a firewall, but it does not include a visualization of the full policy. Firmato includes a user interface that the authors call a "rule illustrator." The rule illustrator presents a firewall policy in a graph in which host-groups are the nodes and rules are indicated by edges. The rule illustrator's graph-based presentation is limited in how much of the policy it can legibly represent on screen without becoming "spaghetti-like" (as the authors note) and it does not explicitly show negative rules. In a field deployment, the authors found a need for more advanced visualization techniques like zooming and/or filtering.

Rode et al. designed Impromptu, an file sharing application [108]. Impromptu includes a visualization-based user interface for specifying file sharing policies. The Impromptu visualization depicts users, files, and actions on a pie chart. Its primary limitation seems to be that it was designed for small groups of users (on the order of 6), and cannot scale to show policies with many users. An evaluation of the Impromptu interface showed that the visualization-based approach held promise for the small-group file-sharing domain; however, the evaluation did not give any evidence as to generalizability of the Impromptu visualization approach.

Lipford et al. designed a user interface for specifying privacy policies in Facebook, a social-networking application [82]. They note that the Facebook site's privacy settings interface was centered around a list of rules and provided minimal feedback on the effects of privacy settings. Lipford et al. designed an "audience view interface" in which Facebook users could view their privacy policies by selecting different audiences (e.g., "friends," "inside network," "outside network") and viewing the profile information that that audience would see. In this respect, Lipford et al. recognize, as we have, the importance of showing policy authors the effects of their policy settings.

Kapadia et al.'s Know system is not a user interface per se, but does address usability for policy-based systems [67]. Know sits within an access control system and provides feedback when access is denied. The feedback can explain why access was denied and how access might be obtained. Although Kapadia et al. discuss what portions of a policy should be revealed in providing feedback, they do not discuss how to design a user interface to present feedback to users.

Lampson was the first to describe access control matrices, in which computer security policies are laid out in a matrix with resources along one axis, principals along the other axis, and the actions allowed those principals for those resources indicated in the matrices' cells [73]. However, Lampson was concerned with describing a security model rather than presenting a user interface, so his matrices were meant more as a pedagogical concept rather than an actual interface to a policy. For example, Lampson did not discuss how his matrices might be scaled to display large policies.

2.2 HCISEC

Our work on policy-authoring interface design is part of the broader usable security movement, also known as HCISEC, to address widespread usability problems in user interfaces for computer security and privacy applications. Here, we discuss how our work fits into the larger themes of HCISEC.

2.2.1 Whitten and Tygar's "problematic properties of security"

Whitten and Tygar, in their seminal work on user interfaces for security applications, laid out five "problematic properties of security" that pose problems for interface designers [139]. Whitten and Tygar's five properties are:

1. The unmotivated user property;
2. The abstraction property;
3. The lack of feedback property;
4. The barn door property; and
5. The weakest link property.

Our work addresses three of these properties:

- *The unmotivated user property.* Security is a secondary goal; most users have a primary goal they wish to complete and do not want to spend their time setting up security and privacy policies. Our work addresses this property by offering interface techniques that allow for fast policy authoring. We do not address how to motivate, educate, or coerce users into performing security tasks; we leave that to others, and merely hope to make the most of the time users are willing to spend authoring policies.
- *The lack of feedback property.* Since a policy is set up to govern access attempts that will happen in the future, there is no natural way for a policy author to receive immediate feedback on whether their policy will allow authorized accesses and prevent undesired accesses. Our work attempts to remedy this situation by providing immediate visual feedback on what decisions a policy will and will not issue.

- *The barn door property.* One undesired access may compromise a resource entirely. Thus, it is essential to get policy settings correct from the start. Our work strongly emphasizes accuracy rate as a metric in evaluation of our policy-authoring interfaces.

2.2.2 Explicit versus implicit policy creation

Policy can be created explicitly through authoring it or implicitly through inferring it. A system administrator manipulating access control lists or a chief privacy officer writing out a company privacy policy are examples of the explicit approach. Granting Excel access to a spreadsheet when a user opens the spreadsheet is an example of the implicit approach. There is no consensus within the HCISEC community on whether one approach is superior; most likely, both approaches are needed to achieve high levels of usability and security. In line with Whitten and Tygar's observation that security is a secondary goal, numerous authors in HCISEC have pointed out that people tend to avoid performing security tasks and even circumvent security mechanisms to accomplish their primary goals [1, 22, 146]. The implication of these works is that designing systems to support implicit policy setting would more likely lead to "better" policies ("better" in the sense of giving only as much access as is needed to accomplish a policy author's goals). Indeed, Yee argues for "security by designation," in which users grant access to resources implicitly in the course of their work when they request that a system carry out an action [147]. Yee gives as examples an application automatically getting access to the CPU when a user executes it or recipients of email attachments automatically receiving access to the attachment because the sender sent it. The Polaris system puts Yee's idea into practice [126].

Our approach addresses explicit policy creation, so it is grounded in the assumption that there is some value in explicit policy creation. If all policy creation and management could be done implicitly, there would be no need for an Expandable Grid policy-authoring interface. However, it is unrealistic to expect that implicit policy creation alone is sufficient in most cases. Explicit policy creation may be needed when responsibility for security is delegated away from end users, as when dedicated system administrators configure access control or firewall policies for end users. Explicit policy creation is needed when a policy is created for communication to others, as when an enterprise authors a privacy policy. Moreover, explicit policy creation provides assurance that a policy is configured the way it is meant to be; implicit policy, since it is not visible, may not provide such assurance. Indeed, Yee agrees that there are situations in which security by designation is insufficient [147] and he promotes the "Principle of Visibility," which calls for a display of relevant policy [146]. Whalen et al. found that people in an office work environment do in fact explicitly set file permissions, and they, like Yee, argue for making access control decisions visible [138]. We maintain that while implicit policy creation is an interesting and valuable approach, there remains a need for good tools for explicit policy creation.

Related to the question of whether policy should be created explicitly or implicitly is the question of whether it should be created proactively or reactively. Proactive policy creation occurs absent any requests for access to resources and is essentially the same as explicit policy creation. Reactive policy creation occurs in response to and at the time of a request (reactive policy creation is different from implicit policy creation in that it requires explicit action on the part of a policy author to approve or deny a request). We designed the Expandable Grid to address proactive policy creation, but some authors question whether proactive configuration is appropriate in some policy-authoring scenarios. In the domain of privacy, for example, Palen and Dourish point out that privacy preferences are a "dynamic response to circumstance," implying privacy preferences are not easily codified into a static policy [97]. Similarly, Lederer et al. warn against emphasizing "configuration" (i.e., proactive policy creation) in designing privacy into interactive systems [75]. However, proactive policy creation is likely needed in most policy domains for the reasons we mentioned above in arguing for explicit policy creation. Some systems, such as the smartphone interface to the Grey system for building access control, offer both proactive and reactive policy creation capabilities [17].

2.3 Cognitive science

Our idea to create a user interface that graphically represents policies to authors is grounded in cognitive theory of external problem representation. Cognitive science has shown us that it is important to provide external representations (text, graphics, charts, etc.) of problems to relieve cognitive burden [118, 96] and that choosing a good external representation of a problem can significantly improve performance on cognitive tasks [12, 95, 143, 144, 151, 149, 150]. Indeed, the entire field of information visualization is predicated on the idea that good external representations of data aid cognitive tasks that are difficult with poor external representations or no external representations, and that they enable cognitive tasks that were impossible before [31]. Our notion that policy authoring will be easier when the user interface allows authors to work directly with effective policy instead of with arcane rules is consistent with Shneiderman's principle of direct manipulation for user interface design [121].

2.4 Information visualization

Information visualization began at the confluence of computer graphics, human-computer interaction, perceptual psychology, and databases [30, 32, 135]. Research in information visualization has produced a wide variety of techniques for displaying large multidimensional datasets on a computer screen and allowing users to quickly pick out patterns in multidimensional data. Here we review those techniques that most closely resemble the Expandable Grid, a two-dimensional interactive table with expandable tree views along its axes.

2.4.1 Interactive tabular visualizations

Rao and Card's Table Lens is a focus+context [33] technique for visualizing large multidimensional data sets in a table[101]. The Table Lens shows a massive data set by showing a few selected rows and columns in full detail while reducing the display size of the remaining rows to show them all in the remaining space. The Table Lens does not include the idea of adding or removing grid cells as a corresponding tree view along one of the axes is expanded or contracted, nor has the Table Lens been adapted specifically to visualizing policy data.

Shneiderman et al. describe Dotfire, an information visualization tool using discrete variables and hierarchical axes to merge searching and browsing (also known as GRIDL (GRaphical Interface for Digital Libraries), that includes a grid with a hierarchically-labeled axis [123]. However, Dotfire only displayed the nodes of a single level of a hierarchy at one time along an axis.

FOCUS is an interactive table (a spreadsheet) that displays product information to help with purchasing decisions [125]. It shows a list of products along the horizontal axis, a hierarchy of features along the vertical axis, and information about which products have which features in the table. It uses an expandable tree view to show the feature hierarchy, and is the first visualization we are aware of to use an expandable tree view along one of its axes. It only has a tree view along one axis, however, and was not used to display policy data.

Other visualization systems built around the idea of spreadsheets whose cells contain images or sub-visualizations rather than simply numerical or textual data include systems by Piersol [98], Levoy [81], Varshney and Kaufman [134], and Chi et al. [59]. Besides being tabular in layout, however, these systems have little in common with the Expandable Grid.

2.4.2 Hierarchical displays

Visualizations that are farther afield include a variety of means for displaying hierarchies such as Johnson and Shneiderman's TreeMaps [66], Lamping and Rao's Hyperbolic Browser [72], Robertson et al.'s Cone Trees [107], and Kumar et al.'s PDQ Tree-browser [71]. These visualizations describe novel ways to display a single hierarchy, but are not concerned with displaying data at the intersection of two hierarchies, as we are.

2.4.3 Security visualizations

Besides the policy-authoring interfaces already reviewed, there have been other efforts to develop information visualization interfaces to help humans manage computer security problems. Conti describes many such systems in his book on security visualizations [36]. Conti covers systems for applications such as viewing network traffic, viewing firewall and intrusion detection logs, and detecting port scans. Stoll et al. designed SESAME, a visualization designed to provide enough information to help end users make security decisions to prevent attacks like phishing, spyware, and bot infections [127]. McLachlan et al. designed LiveRAC, a reorderable matrix visualization of system management data, such as CPU usage and traffic volume, for many network devices [87]. These works do not describe any visualizations of policy, however.

PART II

Preliminary Work

CHAPTER 3

Salmon: Showing Effective Policy

This chapter is largely a reproduction of a paper co-authored with Roy A. Maxion and published in a special issue of the Internation Journal on Human-Computer Studies [85]. We gratefully acknowledge contributions from Fahd Arshad, David Banks, Patricia Loring, and Rachel Roberts.

One locus of vulnerability in a computer system is an undependable user interface—one that does not meet its specification in terms of the speed or accuracy with which users should complete tasks. One reason why some user interfaces fail to meet their speed and accuracy specifications is human error. Researchers have long recognized that human error has causes and manifestations that are similar across all domains of human endeavor, from aviation, to power plant operation, to making a cup of tea [96, 102, 119, 141]. In the domain of software user interfaces, human error leads people off the path of correctly completing a task and on to lengthy task delays or total task failure. There is a need for user-interface designers to understand the common types and causes of human error, and the ways in which they may be prevented. When interfaces are designed to eliminate the conditions that lead people to make mistakes, interfaces will be more dependable, and the applications they serve will be more secure.

One domain in which user-interface accuracy is critically important is computer security. Inaccurate security settings can have a high cost—they can make sensitive data vulnerable, or they can leave an entire system open to attack. Adding to this cost, security problems have what Whitten and Tygar [139] have called the "barn door property"—once a system has had a vulnerability for any length of time, there may be no way to know if the vulnerability has been exploited, so the system will have to be considered compromised, whether it has been or not.

The present work investigates user-interface dependability and human error in the security context of setting file permissions under Microsoft's Windows XP operating system, which uses Microsoft's NT file system (NTFS). NTFS file permissions were chosen as the domain of study because a significant amount of anecdotal evidence suggests that setting NTFS file permissions is a particularly error-prone task for which the consequences of failure can be severe. For example, there is the so-called "Memogate" scandal, in which staffers from one political party on the United States Senate Judiciary Committee stole confidential memos from the opposing party [132]. The memos were stored on a shared NTFS server. The theft was possible in part because an inexperienced system administrator had failed to set permissions correctly on the shared server. As another example, a Windows network administrator at Carnegie Mellon University reports that many users want to share their files so they can access them both at work and at home; they accidentally make their private files accessible to all (several hundred) users on the network, because it is too confusing to set permissions as actually desired [124]. Finally, Microsoft publishes a list of "best practices" for NTFS security that advises users not to use several of the provided features of the NTFS permissions model, such as negative (i.e. deny) permissions and the ability to set permissions on individual files as opposed to folders [89]. The best-practices document states that use of these features "... could cause unexpected access problems or reduce security." Providing access to features which are apparently problematic is bound to lead to errors.

As these anecdotes indicate, setting and checking permissions cannot always be left to expert system administrators—users in many environments need or want to take responsibility for protecting their own files. However, setting file permissions is not an everyday task; it may need to be done only every few weeks or months. Thus, those setting file permissions will often not be expert system administrators; they will be novice or occasional users who, from time to time, want to restrict access to data or to grant access for only a limited number of associates. They will not readily remember arcane details about how to operate a file-permissions-setting interface. The present work adopts the underlying assumption that file-permissions-setting interfaces should accommodate novice and occasional users.

Results of an investigation into and a solution for one type of human error encountered in file-permissions setting interfaces are reported here. First, an existing interface for setting NTFS permissions, the Windows XP File Permissions interface (hereafter abbreviated XPFP), was evaluated in a laboratory user study and shown to have accuracy rates as low as 25% on file-permissionssetting tasks. Errors made by users in the XPFP interface were identified and categorized into types according to an established human-error framework. Goal errors, the failures of users to understand what to do, were identified as the dominant type of error. A primary cause of goal errors, namely poor external representation of task-relevant information, was identified. A design principle, external subgoal support, was proposed to reduce goal errors and was implemented in a new interface, called Salmon, for setting NTFS file permissions. The design principle was evaluated in a laboratory user study identical in design to the XPFP study but employing the new interface. Salmon achieved a success rate of 100% on a task for which XPFP had achieved a 25% accuracy rate, achieved a 94% reduction in the number of goal errors users made on the same task, and achieved a nearly 3× task-performance speed-up in another task, compared to XPFP.

3.1 Objective

The objective of the present work is to understand the causes of user error in user interfaces generally, and file-permissions interfaces in particular. It is a further objective to find a design method that can be applied to new generations of user interfaces so that the same user errors are not encountered again and again and again in future user interfaces.

A specific problem, poor external representation, was observed in the XPFP interface. It was expected that poor external representation would lead to user goal errors, i.e., the incorrect establishment or omission of cognitive goals. A solution was proposed to reduce the occurrence of goal errors. This solution, called external subgoal support (ESS), led to this work's hypothesis:

> Use of ESS in user-interface design reduces the likelihood that users will commit goal errors, thus improving task accuracy rates, and reduces time spent looking for information, thus

reducing time to task completion. The combined improvements to speed and accuracy make the interface more dependable for quick and accurate accomplishment of tasks.

Task success rates, goal-error occurrences, and times to task completion were compared between the XPFP and Salmon studies to determine whether ESS as implemented in Salmon was an effective means of improving success, reducing goal errors, and improving task-completion speed.

3.2 Background and related work

Related work falls into three areas. The first is prior work on usable file permissions and usable access control. The second, a superset of the first, is work in the emerging field of human-computer interaction and security, also known as HCISEC. The third is the traditional literature on human-computer interaction and cognitive science.

File permissions are an instance of the broader area of access control, including evaluation of interfaces for setting file access. Zurko et al. [153] conducted a user study on the Visual Policy Builder, a graphical user interface for specifying access control policies for their Adage system. Good and Krekelberg [55] showed that the Kazaa peer-to-peer file-sharing service's interface misled many users into unintentionally sharing confidential files. Long et al. [83] evaluated a preliminary, paper-based interface for limiting applications' access to system resources. While these three interface evaluations were interesting in their specific task domains, none appear to lead to any conclusion about design principles for security interfaces in a larger context. Other work in usable access control in various domains includes Dewan and Shen [42], Sampemane et al. [114], and Balfanz [11]. With the exception of the Adage project and Long et al., work in this area involves outlining access control models, not evaluating access control interfaces, as the present work sets out to do.

In the broader human-computer interaction and security literature, those who have acknowledged the challenges of designing dependable user interfaces in security-related domains and have proposed principles for better security interface design include Zurko and Simon [154], Adams and Sasse [1], Whitten and Tygar [139], Yee [146], and Besnard and Arief [22]. Such papers propose ideas for making security tasks easier to perform accurately, but do not evaluate their ideas empirically. Whitten and Tygar [140], in another paper, present user-study results to validate a design principle for security interfaces, but their principle, "safe staging," is not related to ESS, nor does it appear to be grounded in a theory of human error.

The traditional human-computer interaction and cognitive science literature contains a substantial amount of material on the importance of external representations in supporting problem-solving tasks [143, 96, 144, 95, 151, 12, 149, 150]. External representations are information displays (e.g., notes on paper, graphics on a monitor, etc.) that reside outside the mind; they complement internal mental representations by providing additional memory capacity, cues to internal processes, and information structures that allow patterns to be easily perceived. Good external representations can facilitate fast and accurate problem solving, while poor external representations can impede it. Mitchell and Miller [92], for example, suggest a methodology for information display design, although their method is intended mainly for industrial domains with expert users, rather than for novice and occasional users.

3.3 Example problem

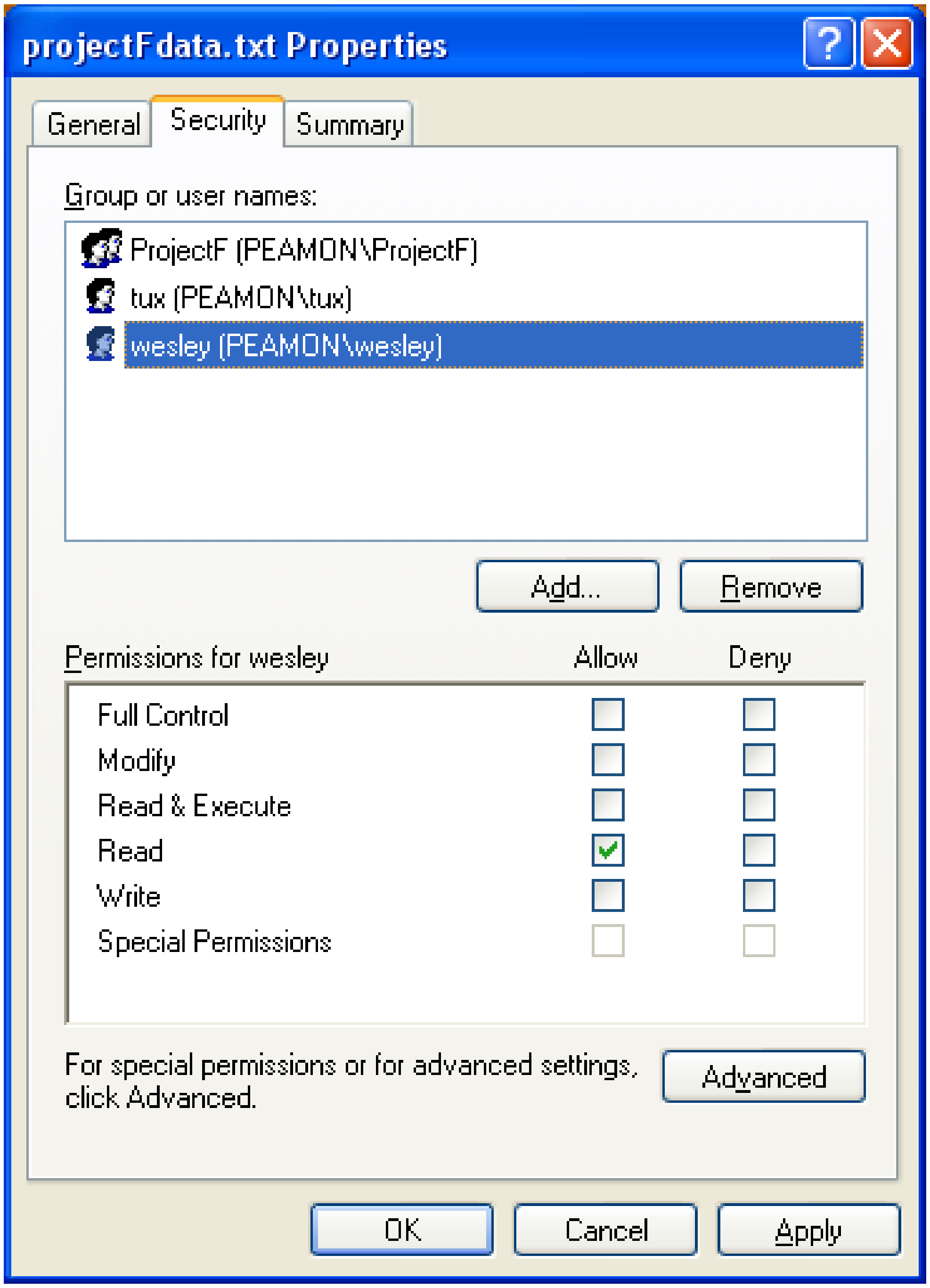
The XPFP interface serves to illustrate the problem with user interfaces that offer poor external representations of task-relevant information. Some background on the NTFS file-permissions model is necessary to fully understand the XPFP interface and the errors it causes.

A computer system using NTFS will be populated with entities and objects. The entities are individual users and groups of users on the system. The objects are the files and folders on the system. NTFS defines 13 atomic permissions (Note that NTFS documentation uses the term special permission where atomic permission is used here. The latter term makes it clearer that these are indivisible permissions, the lowest-level permissions in the system.) that correspond to actions that users can perform on files and folders. The precise meanings of the 13 NTFS atomic permissions are not relevant to this work, but are described in a Microsoft Technet article [88]. For purposes of this work, it is sufficient to note that NTFS permissions can be grouped into five disjoint sets: READ, WRITE, EXECUTE, DELETE, and ADMINISTRATE. (Note that the XPFP interface and NTFS documentation use a different, non-disjoint grouping of the 13 atomic permissions into six composite sets. The disjoint grouping discussed here is the authors' own, and is used for clarity of presentation to those readers not already familiar with the NTFS permissions model.) NTFS uses an Access Control List (ACL) model of file permissions. Under the ACL model, each file and folder in the file system has an associated list of users and groups who have permissions on that file or folder.

For each file and for each permission for that file, each user and group on the file's ACL has a permission value explicitly granted to them for that file and that permission. Permissions are tri-valued, and can take on any of the following values: ALLOW, DENY, or NOTSET; the NOTSET value indicates that neither ALLOW nor DENY has been set. Under the rule of group inheritance, each user also inherits permissions from the groups of which they are members. A user's actual access to a file is computed according to a formula that sums the user's explicit permissions and their inherited permissions according to precedence rules. The precedence rules state that any DENY permission value takes precedence over all other permission values, and any ALLOW permission value takes precedence over any NOTSET permission value; NOTSET acts to deny access by default. Group inheritance leads to the distinction between stated permissions, the explicit permissions granted to each user, and effective permissions, the actual access a user will be allowed according to the combination of stated and inherited permissions.

It is the distinction between stated permissions and effective permissions that makes setting NTFS file permissions a difficult task. The person setting permissions operates directly on the low-level stated-permissions bits, which do not necessarily translate directly into the effective access that will be allowed to system files. Actual access in NTFS is determined by the nuanced formula alluded to above. However, users setting file permissions are ultimately concerned with who can access what, not with low-level bits and nuanced formulas. Thus users need to view the effective permissions in order to evaluate when they have fully completed their primary goal, and when they still have additional work to do to complete their goal.

Casual observation of the XPFP interface (see Figure 3.1) reveals that XPFP does not provide ready access to needed information. Firstly, not all of the NTFS permissions are visible in the main XPFP window. ADMINISTRATE and DELETE permissions are notably absent, and are hidden two screens away. Without ADMINISTRATE and DELETE permissions visible on the main window, some users may not even realize they exist. Secondly, XPFP does not provide the group membership data to indicate what groups individual users belong to. For example, in Figure 3.1, Wesley is in the group ProjectF, but there is no indication of this in the XPFP window. In fact, users need to use an entirely different application to view group membership in Windows XP. Without group membership information, even those few users who understand the group



projectFdata.txt Properties

General | Security | Summary

Group or user names:

- ProjectF (PEAMON\ProjectF)
- tux (PEAMON\tux)
- wesley (PEAMON\wesley)

Add... Remove

Permissions for wesley	Allow	Deny
Full Control	☐	☐
Modify	☐	☐
Read & Execute	☐	☐
Read	✓	☐
Write	☐	☐
Special Permissions	☐	☐

For special permissions or for advanced settings, click Advanced.

Advanced

OK Cancel Apply

Figure 3.1: *A screenshot of the XPFP interface, showing information and functionality for setting permissions bits. Effective permissions are not visible, nor is group-membership information provided, making it virtually impossible for users to verify the completion status of their goals.*

inheritance rules will not be able to figure out whether Wesley is inheriting permissions from ProjectF or not. Finally, the XPFP main window contains the checkboxes necessary for setting the permissions bits that will be used to determine effective permissions, but effective permissions themselves are nowhere to be seen. In fact, XPFP has an effective-permissions display, but it is two screens away, where hardly any non-expert user will find it. The difficulty of accessing essential taskrelevant information in XPFP is likely to lead users into incorrectly determining when they have completed their goals, or to leave them not knowing how to proceed toward task completion. Without a salient external representation of task-relevant information to cue users toward proper behavior, users will be able to complete tasks only if they already have the information stored internally, or if they are fortunate enough to happen upon the information elsewhere. Those who do not have the needed information are likely to commit errors.

3.4 Solution—external subgoal support

External subgoal support (ESS) is an interface design method rooted in cognitive principles regarding the importance of external representations: see Woods and Roth [144], Zhang and Norman [151], Ballard et al. [12], and Zhang [149, 150]. ESS was developed to ensure that all task-relevant information is represented saliently by the user interface, allowing users to check goal completion status and set appropriate subgoals. When users do not set appropriate subgoals, they commit goal errors, one of several specific types of human error. ESS mitigates one cause of goal errors by providing external cues as to what to do. Goal errors are discussed in more detail in Section 3.6.4.

3.4.1 Description

For an understanding of the cognitive causes of goal errors, the present work relies on Pocock et al.'s THEA—Technique for Human Error Assessment [99]. THEA was developed to analyze a user-interface design for areas of potential user difficulty without conducting costly user studies. It includes an error framework which is based on Norman's well-known seven-stage execution-evaluation model of human information processing [96]. THEA condenses Norman's seven stages down to four stages of information processing during which human error can occur. These four stages are the combination of perception, interpretation, and evaluation; goal formulation; plan formulation; and action execution. According to the Norman/THEA models, human information processing starts with a problem, the primary goal, and proceeds in the following loop:

1. Perceive and interpret information from the environment, and evaluate whether the problem is solved.
2. If the problem remains unsolved, formulate a subgoal, according to perceived information, for solving all or part of the problem; if the problem is solved, exit the loop.
3. Formulate a plan to achieve the subgoal.
4. Execute the actions in the plan.

Goal errors occur when the second step goes wrong. If the perceived information consulted in the second step is incorrect or is misinterpreted, the wrong subgoal may be set. If the wrong information is used to check whether the problem has been solved in the second step, either an unnecessary subgoal may be added (if the problem is assumed unsolved when it is already solved) or a necessary subgoal may be omitted (if the problem is assumed solved when it is not). Thus, the availability of information to check progress toward the primary goal is critical to correct selection of subgoals.

If incorrect, misleading, or missing information causes goal errors, the logical solution is to make the necessary information available in a correct, easily interpretable form. Indeed, researchers in cognitive science have shown the significant effects that external (i.e., external to the human problem-solver) problem representations can have on human performance: see Woods and Roth [144], Zhang and Norman [151], Ballard et al. [12], and Zhang [149, 150]. External representations not only serve as memory aids, but also play a central role in shaping cognitive behavior [151]. Some representations of a problem lead to incorrect or slow problem-solving behavior, while other representations facilitate correct and efficient problem-solving behavior. Section 3.8 discusses how each of the user interfaces tested in the present study influences users to engage either more or less in activities like information-gathering, based on the information representations made available to the users.

3.4.2 Design method

A prerequisite for implementing ESS is a careful task analysis. Kirwan [70], an excellent reference on how to perform task analyses, describes the Hierarchical Task Analysis (HTA) method, which includes a convenient representation of the results of a formal task analysis. An HTA represents the task as a hierarchy of goals and the operations that are needed to achieve them. At the root of an HTA hierarchy is a primary goal to be accomplished. Beneath the root are nodes that represent the subgoals necessary to achieve the primary goal, and each subgoal may have a tree of subgoals beneath it. At the leaf nodes of the hierarchy are the actionable operations necessary to achieve each of the lowest-level subgoals. Once the HTA is complete, and its corresponding hierarchy of subgoals has been created, the method for designing according to ESS can begin. It proceeds in two phases as shown below. The Salmon interface, shown in the next section, was designed according to this method.

- Phase 1—Identify information required.
    1. For each goal, starting with the primary goal and proceeding through all subgoals in the HTA, identify the information a user will need:
        (a) To determine when the goal has been completed.
        (b) To set the subgoals beneath the goal.
    2. For each operation at the leaf nodes of the HTA, determine what information will be needed to execute the operation. This is usually:
        (a) Procedural knowledge—information about how to execute the operation.
        (b) Declarative knowledge—any parameters that must be supplied to the operation (e.g., the name of a user being added to an ACL).
- Phase 2—Provide Phase-1 information in the interface.
    1. Incorporate, in the designed interface, an accurate, clear, and salient representation of the necessary information, as determined by the above steps. (External representation design is a large topic and is not covered in detail here; see Card [30] or Woods [144] for more on this topic.)

3.4.3 Salmon interface with ESS

The Salmon interface (see Figure 3.2) was designed according to the ESS method. Phase 1 of the ESS method identified the following information as necessary for establishing the right subgoals and executing the right operations in a file-permissions interface:

1. The full list of 13 atomic permissions, with no permission values hidden.
2. Stated permissions for all users and groups on the ACL.
3. Group membership data, and how the data combine to form a user's effective permissions.
4. Effective permissions for all users on the ACL.

The Salmon main window contains all of this information. The window is split into two panes, upper and lower. In the upper pane, the 13 atomic permissions are listed across the top; immediately below them are the checkboxes necessary for setting the stated permissions for each user or group on the ACL. In the lower pane is an effective permissions display. For any individual user selected, the effective-permissions display shows the groups of which that user is a member, the permissions inherited from those groups, the user's individual stated permissions, and the combination of all these permissions, i.e., the user's effective permissions.

3.5 User study and experimental method

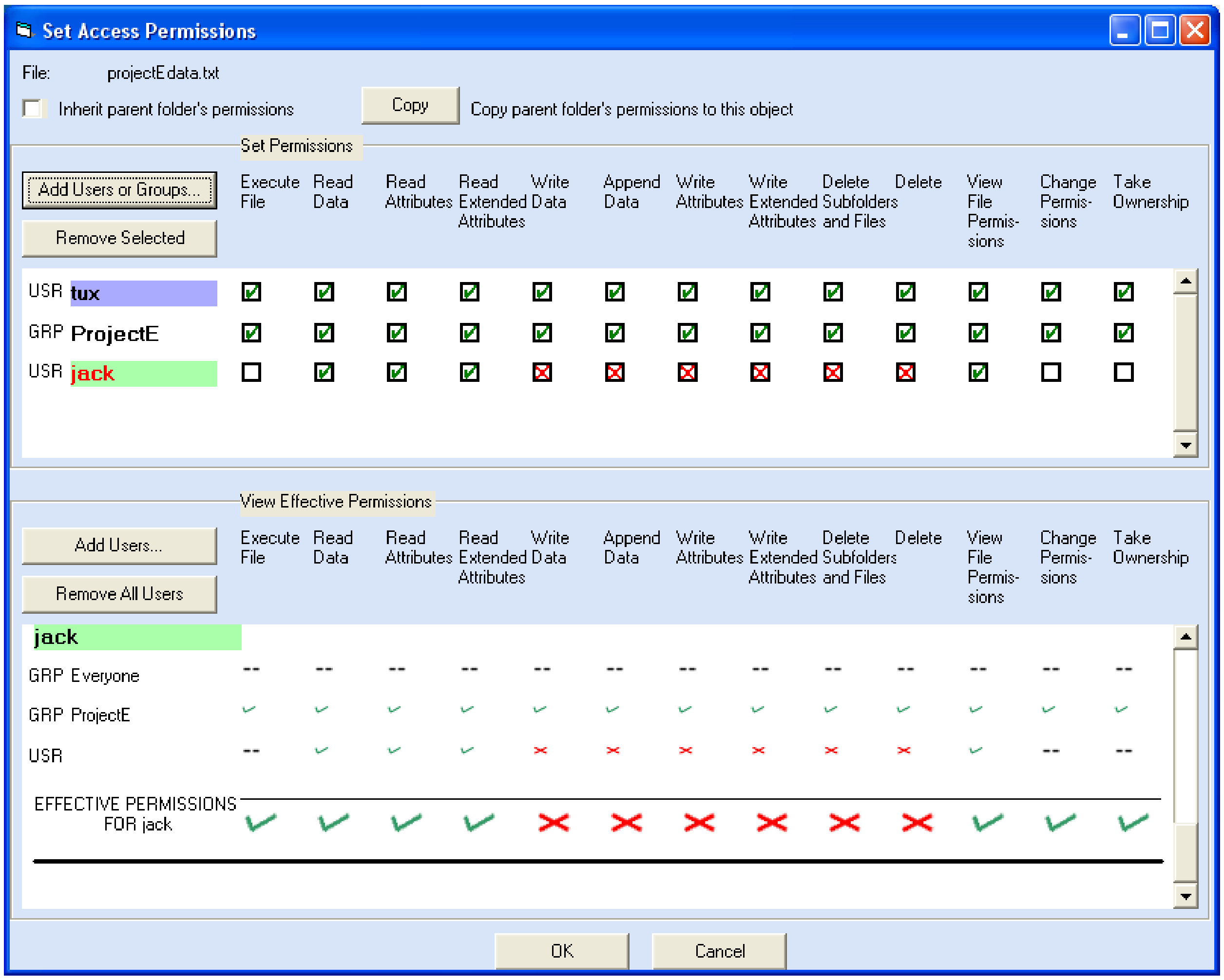
A laboratory user study was conducted to observe and document errors in filepermissions- setting tasks. Two user interfaces were compared: XPFP and Salmon, a new interface for setting file permissions, designed in accordance with the principles of external subgoal support.

3.5.1 Participants

Twenty-four students and research staff at Carnegie Mellon University voluntarily participated in the study. Participants were recruited randomly. All participants' academic backgrounds were in science and engineering disciplines, and all were daily computer users. While a few usually used UNIX-based computer systems in their daily work, all had at least some experience using Windows, with 21 out of 24 claiming they used Windows at least a few times a week. Nineteen reported having some experience setting file permissions, on Windows or another operating system, while 5 reported having no experience setting file permissions whatsoever. All but four reported setting file permissions a few times a month or less.

3.5.2 Apparatus

All participants solved a series of permission-setting tasks on a computer running Windows XP, Version 2002, Service Pack 1, using either the XPFP or Salmon permission-setting user interface. Timestamped screen video and mouse and keyboard actions were recorded with a software tool developed for user study data collection. Participants' initial and final permissions settings were recorded for each task instance.



Set Access Permissions

File: projectEdata.txt

☐ Inherit parent folder's permissions [Copy] Copy parent folder's permissions to this object

Set Permissions

[Add Users or Groups...] [Remove Selected]

	Execute File	Read Data	Read Attributes	Read Extended Attributes	Write Data	Append Data	Write Attributes	Write Extended Attributes	Delete Subfolders and Files	Delete	View File Permissions	Change Permissions	Take Ownership
USR tux	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑
GRP ProjectE	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑
USR jack	☐	☑	☑	☑	☒	☒	☒	☒	☒	☒	☑	☐	☐

View Effective Permissions

[Add Users...] [Remove All Users]

jack	Execute File	Read Data	Read Attributes	Read Extended Attributes	Write Data	Append Data	Write Attributes	Write Extended Attributes	Delete Subfolders and Files	Delete	View File Permissions	Change Permissions	Take Ownership
GRP Everyone	--	--	--	--	--	--	--	--	--	--	--	--	--
GRP ProjectE	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
USR	--	✓	✓	✓	✗	✗	✗	✗	✗	✗	✓	--	--
EFFECTIVE PERMISSIONS FOR jack	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✓	✓	✓

[OK] [Cancel]

Figure 3.2: *The Salmon interface (screenshot) was designed to provide external subgoal support by having all 13 atomic permissions be settable in the upper pane, and by showing effective permissions in the lower pane.*

3.5.3 Task descriptions

To simulate real permission-setting conditions, a hypothetical scenario was designed in which the participant worked in a generic "organization," shared her computer with other workers in the organization, and had to restrict access to the files and folders on her computer. On the laboratory Windows XP machine, the hypothetical organization's computer environment was created and populated with individual users, groups (containing users), files, and folders. The environment included 27 individual users, named for each letter of the alphabet (Ari, Bill, Catherine, Dave, Evelyn, etc.), plus one user named Tux, which represented the participant her/himself. The environment also included 6 groups named ProjectA through ProjectF, each of which contained 6 members drawn from the 27 users. No group contained another group as a member. There were also files and folders on which participants were to set permissions.

Participants were given three permission-setting tasks, called Jack, Wesley and Tux, and one training task called the Hakim task. The Hakim (training) task simply required the participant to add a user to the ACL, and to give that user READ permission. This training task gave participants experience with the mechanics of adding users and setting permissions—actions which were common to the experimental tasks.

The Jack and Wesley permission-setting tasks were chosen because they involved group inheritance, a feature of the NTFS permissions model that was expected to lead to a great deal of user error. These two tasks required participants to set permissions on a text file so that users Jack or Wesley could read the file, but not change it. The Tux task was chosen to determine how easily participants could find the DELETE permission checkbox if it were hidden, as it is in the XPFP interface, as opposed to being immediately visible, as it is in the Salmon interface. The Tux task required participants to set permissions on a text file so that Tux would not be able to delete the file by accident.

The task statements for the four tasks are shown below:

- *Hakim (training) task*: You (username: tux) have just created the folder Stuff for Hakim, so that you can share private data with your friend Hakim (username: hakim). Set permissions on the folder so that Hakim will be able to read anything you put in the folder. Make sure no one else can read anything in the folder. i

- *Jack task*: The group ProjectE is working on projectEdata.txt, so everyone in ProjectE can read, write, or delete it. Jack (username: jack) has just been reassigned to another project and must not be allowed to change the file's contents, but should be allowed to read it. Make sure that effective now, Jack can read the file projectEdata.txt, but in no way change its contents.

- *Wesley task*: The group ProjectF is working on projectFdata.txt, so everyone in ProjectF can read, write, or delete it. Wesley (username: wesley) has just been reassigned to another project and must not be allowed to change the file's contents, but should be allowed to read it. Make sure that effective now, Wesley can read the file projectFdata.txt, but in no way change its contents.

- *Tux task*: You (username: tux) have a checkbook-balancing program that writes to a file called myCheckbook.dat. You do not want to accidentally delete this file. Deny yourself the permission to delete it. Of course, you want all other permissions to remain unchanged.

In the Wesley and Jack tasks, there was one group that was already on the ACL for the file, and the operative individual user was a member of that group. The difference between the Wesley and Jack tasks was that in the Wesley task, Wesley was inheriting READ and WRITE permissions from ProjectF, but not

ADMINISTRATE permission, while in the Jack task, Jack was inheriting READ and WRITE as well as ADMINISTRATE permissions from ProjectE.

The simple solution to the Wesley task was to add Wesley to the ACL and explicitly deny him WRITE permission; he was already allowed READ permission from ProjectF. However, this simple solution would not work for Jack, since Jack was inheriting ADMINISTRATE permission as well as READ and WRITE permission. If Jack was denied WRITE permission, but not explicitly denied ADMINISTRATE permission, he would have been able to restore his WRITE permission. The task statement presented to users did not mention this nuance; it was left to the interfaces to provide the cues needed to understand that Jack's ADMINISTRATE permission had to be removed.

3.5.4 Rules for completing tasks

Participants were asked to abide by certain rules to ensure as realistic an environment as possible without compromising the experimental comparison between XPFP and Salmon. First, they were allowed to check group memberships in the XP Computer Management application, which is a separate application from the file-permissions interfaces. However, in order to prevent users from completing tasks without using the file-permissions interfaces, they were asked not to use the Computer Management application to change group memberships. Thus, users were not allowed to complete the Wesley and Jack tasks by removing Wesley or Jack from their respective groups. Second, to compensate for the first restriction, participants were told that if a task statement did not explicitly mention a given user, any permission setting was permissible for that user. Thus, participants could change the group ProjectE's or the group ProjectF's permissions and not be concerned about the effects on members of those groups besides Wesley or Jack. Finally, users were allowed to access a selected set of Windows Help files that pertained specifically to file permissions, but they could not browse the full set of Help files at random.

3.5.5 Procedure

Participants were assigned randomly to use either the XPFP or Salmon interface. Each participant used only the single assigned interface for all tasks; no one used both interfaces. Twelve participants were assigned to each interface. Participants were asked to "think aloud" throughout the course of the experiment, and were instructed in thinking aloud according to directions adapted from Ericsson and Simon [44]. Participants were shown how to view system users, groups, and group memberships using the Computer Management interface and how to access the limited set of Windows Help files. Participants were given no instruction in using the XPFP or Salmon interfaces. The experimenter brought up the interface the participant was to use. Task statements were presented in text in a Web browser, and remained available to the participant throughout the task. All participants were given the same training task first, after which the presentation order of the remaining tasks was counterbalanced among participants using a full factorial design. Participants were given 8 minutes to complete each task. After each task was completed, participants were asked to rate their confidence on a 1-7 scale (7: very confident) that the task had been completed correctly.

3.6 Analytical procedures

This section presents the procedures for data analysis, the results of which are presented in Section 3.7. Data were logged from user sessions automatically by custom software. Separate, but consistently timestamped, files were kept for keyboard, mouse, video and verbal protocols (digital video and audio). These data sets needed to be analysed to recover task-relevant information regarding speed (timing), accuracy (task success or failure), and errors (discrete actions, classification of action as error or non-error, and classification of errors into one of four types). Measuring elapsed times of events is straightforward, and will not be discussed in detail. Other analytical aspects of the experiment are discussed below.

3.6.1 Accuracy—determining task success or failure

Accuracy was measured in terms of whether or not the participant succeeded in correctly executing the task. To determine task success or failure, participants' final permission settings were examined. For the Wesley and Jack tasks, a task instance was deemed successful if the individual (Wesley or Jack) had effective permissions allowing him READ permission and denying him WRITE and ADMINISTRATE permissions. A Wesley or Jack task instance was deemed a failure if the individual had effective permissions denying READ permission, or allowing WRITE and/or ADMINISTRATE permission. EXECUTE and DELETE permissions and all permissions for other entities were ignored. For the Tux task, a task instance was deemed successful if Tux had effective permissions denying DELETE and allowing both READ and WRITE permissions. A Tux task instance was deemed a failure if Tux had effective permissions allowing DELETE, or denying READ or WRITE permission.

3.6.2 Actions

Actions need to be defined so that user data can be divided into discrete units for error analysis. An action is defined as any change to the ACL, namely adding an entity to or removing an entity from the ACL, or altering the permissions of an entity already on the ACL.

3.6.3 Classifying actions as errors

An action is classified as an error if it is discrepant with a model of correct actions. A model of correct actions for each task was constructed by HTA [70]. An example HTA for the Jack task is shown in Figure 3.3; HTAs for other tasks were similarly constructed. The HTA for each task consists of the goals, plans and actions required to complete the task.

Each discrepancy between user actions and HTA model actions was classified as an error of commission, an error of omission, or a non-error. A user action was an error of commission if it was unnecessary according to the HTA. It was an error of omission if it was a necessary action according to the HTA, but the user failed to complete it. Non-errors included user actions that matched actions in the HTA and unnecessary but innocuous actions, such as changing permissions in an interface to "see what happens" and then changing them back.

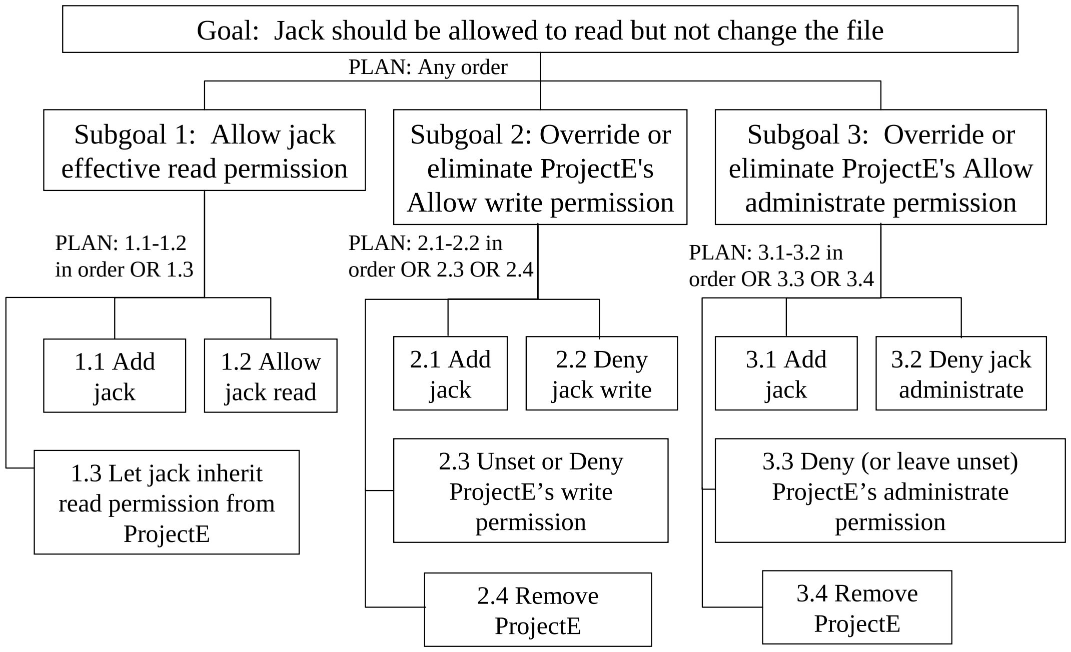

Goal: Jack should be allowed to read but not change the file
PLAN: Any order
Subgoal 1: Allow jack effective read permission
Subgoal 2: Override or eliminate ProjectE's Allow write permission
Subgoal 3: Override or eliminate ProjectE's Allow administrate permission
PLAN: 1.1-1.2 in order OR 1.3
PLAN: 2.1-2.2 in order OR 2.3 OR 2.4
PLAN: 3.1-3.2 in order OR 3.3 OR 3.4
1.1 Add jack
1.2 Allow jack read
1.3 Let jack inherit read permission from ProjectE
2.1 Add jack
2.2 Deny jack write
2.3 Unset or Deny ProjectE's write permission
2.4 Remove ProjectE
3.1 Add jack
3.2 Deny jack administrate
3.3 Deny (or leave unset) ProjectE's administrate permission
3.4 Remove ProjectE

Figure 3.3: *Hierarchical task analysis (HTA) for the Jack task.*

3.6.4 Classifying errors by type

Actions that were classified as errors were further classified according to error type: goal, plan, action, and perception errors. These four types were drawn from Pocock et al.'s THEA which describes four stages of human information processing [99]. There are many other frameworks that could have been used to classify human error. THEA was chosen because it was specifically designed for evaluating user interfaces, and because of its grounding in the familiar work of Norman [96].

Goal errors, the focus of the present work, are described forthwith. Similar criteria were used to classify action, plan, and perception errors. An error was classified as a goal error if it was either: (1) an error of commission that was due to the user's establishing a wrong subgoal; or (2) an error of omission that was due to the user's failure to establish a necessary subgoal. An example of a common goal error from the Wesley task was a user failing to explicitly deny Wesley WRITE permission. In the Jack task, both failing to explicitly deny Jack WRITE permission (omitting Subgoal 2 in Figure 3.3) and failing to explicitly deny Jack ADMINISTRATE permission (omitting Subgoal 3 in Figure 3.3) were goal errors. In the Tux task, denying Tux WRITE permission was a goal error.

3.7 Results

This section presents the results of the study, including details of speed, accuracy and propensity for mitigating errors for each of the two interfaces under scrutiny. The results show that Salmon performed better than XPFP did.

3.7.1 Speed

Figure 3.4 shows the average task completion times for each of the two interfaces and three tasks. The solid bars show times for all participants, whether they succeeded or failed in the task; the striped bars show times only for participants who completed the tasks accurately. For Wesley and Jack, these results are of interest, because they show that the success of Salmon users was not due to having spent more time on task. In fact, those who completed the tasks successfully took less time using the Salmon interface. The difference between times for the two interfaces is not statistically significant (one-sided t-test for Wesley: $t = .3942$, $df = 14.39$, $p = .3496$; for Jack: $t = .8973$, $df = 9.367$, $p = .1961$). For the Tux task, successful Salmon users spent, on average, significantly less time ($M = 60.8$s, $SD = 23.0$) than the successful XPFP users did ($M = 178.0$s, $SD = 108.6$). A one-sided t-test showed this difference to be statistically significant at the .05 level ($t = 3.183$, $df = 8.538$, $p = .006$).

3.7.2 Accuracy

Table 3.1 shows the percentage of participants who successfully (accurately) completed the tasks on the XPFP and Salmon interfaces. In testing whether or not Salmon gives superior performance in terms of accuracy (measured as task completions), a null hypothesis was posited: the proportion of task completions under Salmon was less than or equal to the proportion under XPFP; the alternative hypothesis is that the proportion for Salmon was greater. To test whether two sample proportions are equal, let $\hat{p}_S$ be the proportion of

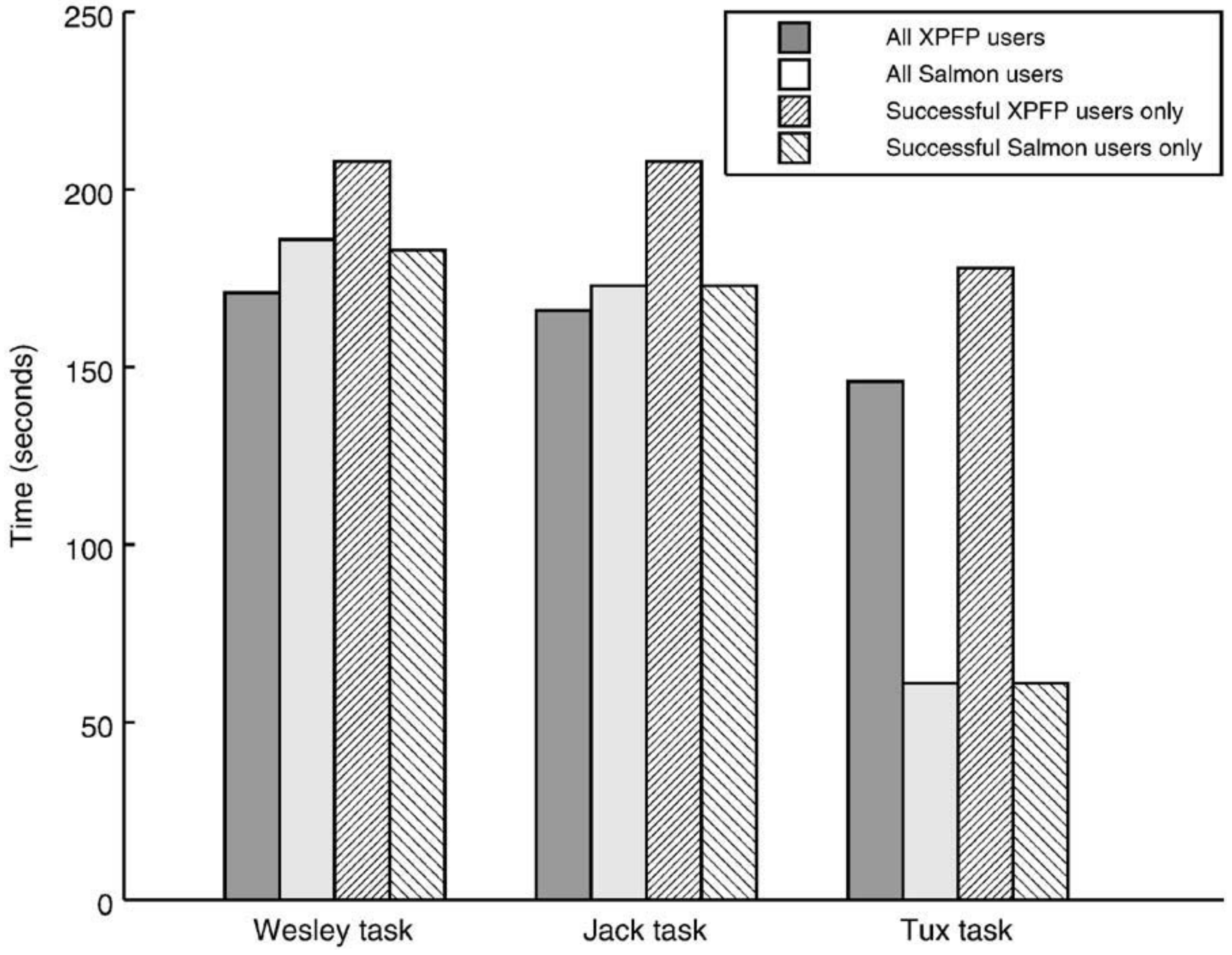


Figure 3.4: *Average time to complete Wesley, Jack, and Tux tasks. Successful Salmon users took less time on average than successful XPFP users did, suggesting that Salmon's accuracy gains were not due simply to a speed-accuracy tradeoff.*

Table 3.1: *Percent of accurate task completions for Jack and Wesley tasks on XPFP and Salmon interfaces. Salmon outperformed XPFP in both tasks.*

	XPFP	Salmon
Wesley	58%	83%
Jack	25%	100%
Tux	75%	100%

completions for Salmon and let $\hat{p}_X$ be the proportion for XPFP. The test statistic is $t_s = \frac{\hat{p}_S - \hat{p}_X}{\sqrt{\overline{p}(1-\overline{p})(\frac{1}{n_S}+\frac{1}{n_X})}}$, where $\overline{p}$ is the total number of completions divided by the total number of subjects (e.g., for Jack, there were 15 (3+12) completions by 24 (12+12) subjects; thus $\overline{p} = \frac{15}{24}$).

A one-sided z-test was done for the equality of proportions. Since the null hypothesis specifies equality, one uses the pooled estimate of the standard error of the difference between the two proportions to calculate the likely size of the chance error in estimating the true difference between the proportions. The test statistic, t_s, for the results of the Jack task was 3.795, which on referral to a z-table has significance probability less than .0001. Thus the null hypothesis can be strongly rejected; for the Jack task the Salmon interface is superior. Salmon is also superior for the Tux task, with a test statistic of 1.852 and significance probability .032; the Wesley task test statistic of 1.347 is weakly significant, with a significance probability of .089.

Recall that participants were asked to state their confidence in their work. Curiously, there was no significant difference between interfaces in participants' confidence ratings, on any task, as to whether tasks were completed correctly or not. XPFP users gave slightly higher confidence ratings than Salmon users for the Wesley (XPFP: $M = 5.58$, $SD = 1.56$; Salmon: $M = 4.83$, $SD = 1.53$) and Jack (XPFP: $M = 5.58$, $SD = 1.39$; Salmon: $M = 5.17$, $SD = 1.64$) tasks, and gave slightly lower confidence ratings for the Tux task (XPFP: $M = 5.50$, $SD = 1.45$; Salmon: $M = 6.17$, $SD = 1.11$). One-sided, unpaired t-tests show Wesley ($t = 1.19$, $df = 21.99$, $p = .88$); Jack ($t = .67$, $df = 21.36$, $p = 0.75$); and Tux ($t = 1.26$, $df = 20.66$, $p = 0.11$) with no significant differences between XPFP and Salmon with respect to user confidence. This is interesting in view of the very different actual outcomes; that is, Salmon facilitated much better success than XPFP did, and yet the perception of participants regarding their success was about the same. This bears investigation, but is not further addressed here.

3.7.3 Errors

Table 3.2 shows results of the error analysis for the three tasks; XPFP dominated in terms of total errors and goal errors. For the Wesley task, 5 of 9 XPFP-user errors were goal errors; 1 of 4 Salmon-user errors was a goal error. For the Jack task, 15 of 16 XPFP-user errors were goal errors; 1 of 6 Salmon-user errors was a goal error. For the Tux task, 2 of 3 XPFP user errors were goal errors; Salmon users made no errors. In testing whether or not Salmon gives superior performance in terms of reducing the likelihood of goal errors, a null hypothesis was posited: the proportion of goal errors under Salmon was greater than or equal

Table 3.2: *Count of errors by type for Wesley and Jack tasks on XPFP and Salmon interfaces. Users made significantly fewer goal errors with Salmon than with XPFP on the same tasks.*

	Goal	Plan	Action	Perception
Wesley				
XPFP	5	1	0	3
Salmon	1	2	1	0
Jack				
XPFP	15	0	0	1
Salmon	1	3	2	0
Tux				
XPFP	2	1	0	0
Salmon	0	0	0	0

to the proportion under XPFP; the alternative hypothesis is that the proportion for Salmon was smaller. As described in Section 3.7.2, a one-sided z-test was done for the equality of proportions. Since the null hypothesis specifies equality, one uses the pooled estimate of the standard error of the difference between the two proportions to calculate the likely size of the chance error in estimating the true difference between the proportions. The test statistic, ts, for the results of the Wesley task was -1.885, which on referral to a z-table has a significance probability of .0297. Thus one can reject the null hypothesis at the .05 level; the Salmon interface is superior for Wesley. For the Jack task, $t_s = -3.312$, and the z-table significance probability is .0005, which is highly significant. For the Tux task, $t_s = -1.477$, and the z-table significance probability is .0698, which is mildly significant. It can be concluded that in terms of mitigating goal errors, the Salmon interface performs better than the XPFP interface does.

3.8 Discussion

A straightforward comparison of accuracy results shows that Salmon outperformed XPFP in all three tasks. The causes for XPFP's poor performance and Salmon's improved performance can be inferred from the error classification data, which shows a substantial reduction in goal errors amongst Salmon users. For some insight into whether Salmon's improved external representation was responsible for the improved performance, users' protocols were analysed to determine the amount of time spent on each of 10 high-level behaviors. The analysis implies that Salmon's external representation of task-relevant data leads users more readily toward finding the information they need.

3.8.1 Goal errors in XPFP

Goal errors were common amongst XPFP users and led to high failure rates. The preponderance of goal errors observed in XPFP is not surprising; as noted in Section 4, the XPFP interface lacks information vital to the file-permissions problem solver (see Figure 3.1). In the main XPFP window, there is no display of effective permissions, no way to access group membership information, and no indication of the existence of ADMINISTRATE permissions. Without effective permissions, progress toward the primary goal cannot be checked accurately. Many users, unaware of the distinction between stated and effective permissions, used the stated permissions to determine whether the primary goal had been completed. For example, during the Wesley task, many users saw the XPFP window in the state shown in Figure 3.1. From this display, it appeared to many users that Wesley was allowed READ permission because his ALLOW-READ checkbox is checked, but not allowed WRITE permission, because his ALLOW-WRITE checkbox is not checked. They did not realize that he has effective WRITE permission from ProjectF. Users hence were misled into thinking they had completed the task, even though they had not.

3.8.2 Fewer goal errors in Salmon

The improvement in task-completion successes and the dramatic reduction in goal errors achieved in the Salmon study can be accounted for primarily by the use of ESS in the design of the Salmon interface. Users of Salmon were able to track their progress and identify the subgoals necessary to complete tasks using the effective permissions display, which provided an external representation supporting subgoaling. Although Salmon's design contains numerous superficial changes from the XPFP design (such as different fonts, labels, icons, colors, and layout), observation of participants' protocols strongly suggests that it was the effective permissions display that led users to formulate the correct goals. For example, one Salmon participant, about to commit an incorrect solution to the Jack task, said, "I see Jack over here now [pointing to Jack's stated permissions], and he doesn't have any access rights... Oh, wait! Jack has access rights over here [pointing to Salmon's effective permissions display]." After noticing the effective permissions display, the participant was able to correctly complete the task. In contrast, one typical XPFP participant, looking at Wesley's stated permissions in the XPFP window as shown in Figure 3.1, said, "And apparently his permissions are just READ. That's what we want." He had not explicitly denied WRITE permission to Wesley, and implemented an incorrect solution. In the absence of correct information to confirm that the task was complete, the participant used incorrect information, the stated permissions, to "confirm" that he had correctly completed the task. As opposed to the XPFP interface, Salmon's external representation enables the user to avoid having to build and maintain an internal mental representation of a complex permission-setting structure. The Salmon display provided the necessary goal cues by holding the representation externally, making it more accessible to the user.

3.8.3 Less time searching for information in Salmon

To determine in more detail how users spent their time, and to address the hypothesis that Salmon users' improvement was due to improved external representations, users' protocols were analysed for time spent on specific behaviors. This kind of analysis reveals how an interface shapes users' behavior, by drawing attention to activities where users spent the bulk of their time. Ten behaviors were identified, and each user protocol was segmented into portions of time spent on each of the 10 behaviors. Objective criteria determined whether a given segment of protocol matched a particular behavior. For example, a "Read task" behavior was identified any time a user had the Web browser with the task statement in the foreground. As another example, a "Learn interface" behavior was identified as the time between first reading the task statement and starting to enact a plan, or any action that followed a user statement like "Let's see what this does." Figure 3.5 shows the average amount of time users spent on each of these ten behaviors for the Wesley task; results for the Jack task are similar, and are not shown. Figure 3.6 shows the average amount of time that users spent on each of these 10 behaviors for the Tux task. Each set of four bars shows, from left to right, the average times for all XPFP users, all Salmon users, successful XPFP users only, and successful Salmon users only. Descriptions and discussion of each of the 10 behaviors is included below.

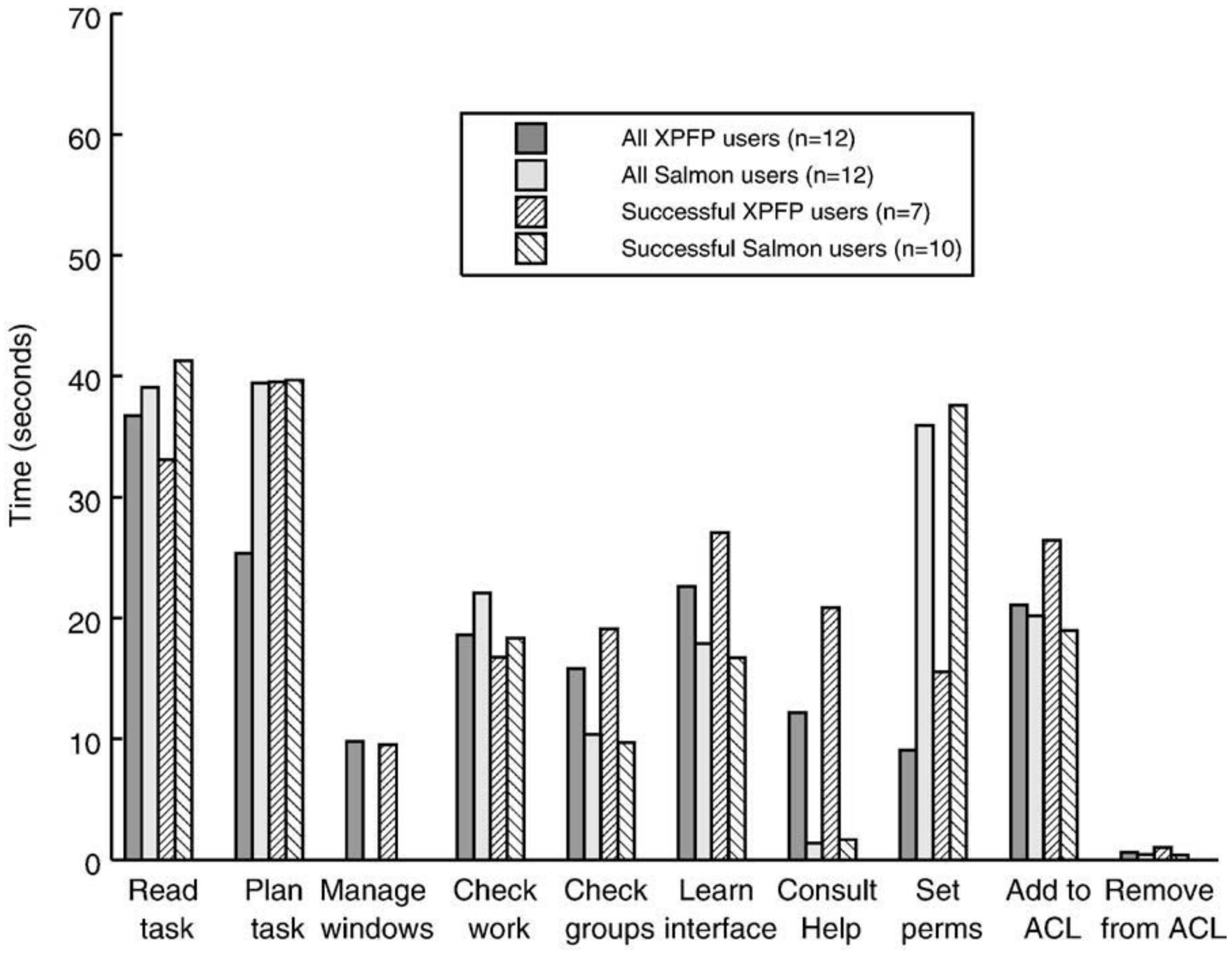


Figure 3.5: *Average time spent on each of 10 behaviors identified in the Wesley task by all XPFP and Salmon users combined, whether successful or not, and only successful XPFP and Salmon users. Salmon users spent less time on information-gathering behaviors like check-groups, learn-interface, and consult-help.*

- *Read task*: This behavior consists of reading the task statement in the Web browser. Since the task statements were presented in the same manner to all users, it is to be expected that users of both interfaces would take about the same amount of time, on average, to read them and refer to them while working.

- *Plan task*: This behavior is assumed to occur when the user is not engaged in any of the other nine behaviors on this list; it consists of thinking and trying to formulate a plan. Notice that, for the Wesley task, successful XPFP users and Salmon users spent more time planning than did all XPFP users. This can be explained by noting that those XPFP users who failed to complete the Wesley task typically failed by omitting a subgoal, and hence finished more quickly than those who successfully completed all subgoals. Notice, too, that XPFP users in the Tux task spent more than twice as much time planning, since without a visible DELETE permission-setting checkbox, it was harder to decide what to do.

- *Manage windows*: The manage-windows behavior includes the activities of moving, opening, closing, minimizing, or otherwise manipulating interface windows. Time spent on the dialog windows for Check groups, Read task, Consult Help, and Add to ACL behaviors is not included within managewindows. Salmon users did not exhibit managing-windows behavior, since Salmon only has one interface window, while XPFP has as many as four. Consequently, it takes XPFP users longer to navigate through the multiple windows to find information. Salmon does take up more space on the screen, however, in order to display all the relevant information in one window.

- *Check work*: This behavior consists of reviewing the interface to confirm that permissions look correct; it occurs after setting permissions and before ending the task or revising work. It is interesting to note that although users spent roughly the same amount of time checking work on both interfaces for all three tasks, there were still more failures amongst those using XPFP. This is likely because the information needed to check work, the effective permissions, is not visible in XPFP, so users employ the wrong information, the stated permissions, to check their work.

- *Check groups*: This behavior consists of checking group membership with the XP Computer Management interface. XPFP does not display group membership information, so it requires users to view the separate Computer Management application to check group membership. Salmon's lower pane displays group information, so it is not necessary to go to another application to check group membership. (Some users did anyway, because they had not yet noticed that the information was available in Salmon). Unsurprisingly, Salmon users spent less time checking group memberships than did XPFP users in the Wesley task. Since groups were not relevant to the Tux task, no users spent time checking groups.

- *Learn interface*: This behavior consists of either looking over the interface at the beginning of the task to see what can be done with it, or experimenting with the interface by clicking on interface elements whose function is unknown or confusing. Novice users spend time surveying a new interface and testing out its functionality to see how it works. Often, when at an impasse, users may test an unfamiliar interface function to see if it helps them accomplish their goal. Since Salmon was a completely new interface, while some users had already used the XPFP interface before participating in the study, it was expected that users would spend more time learning the Salmon interface. This

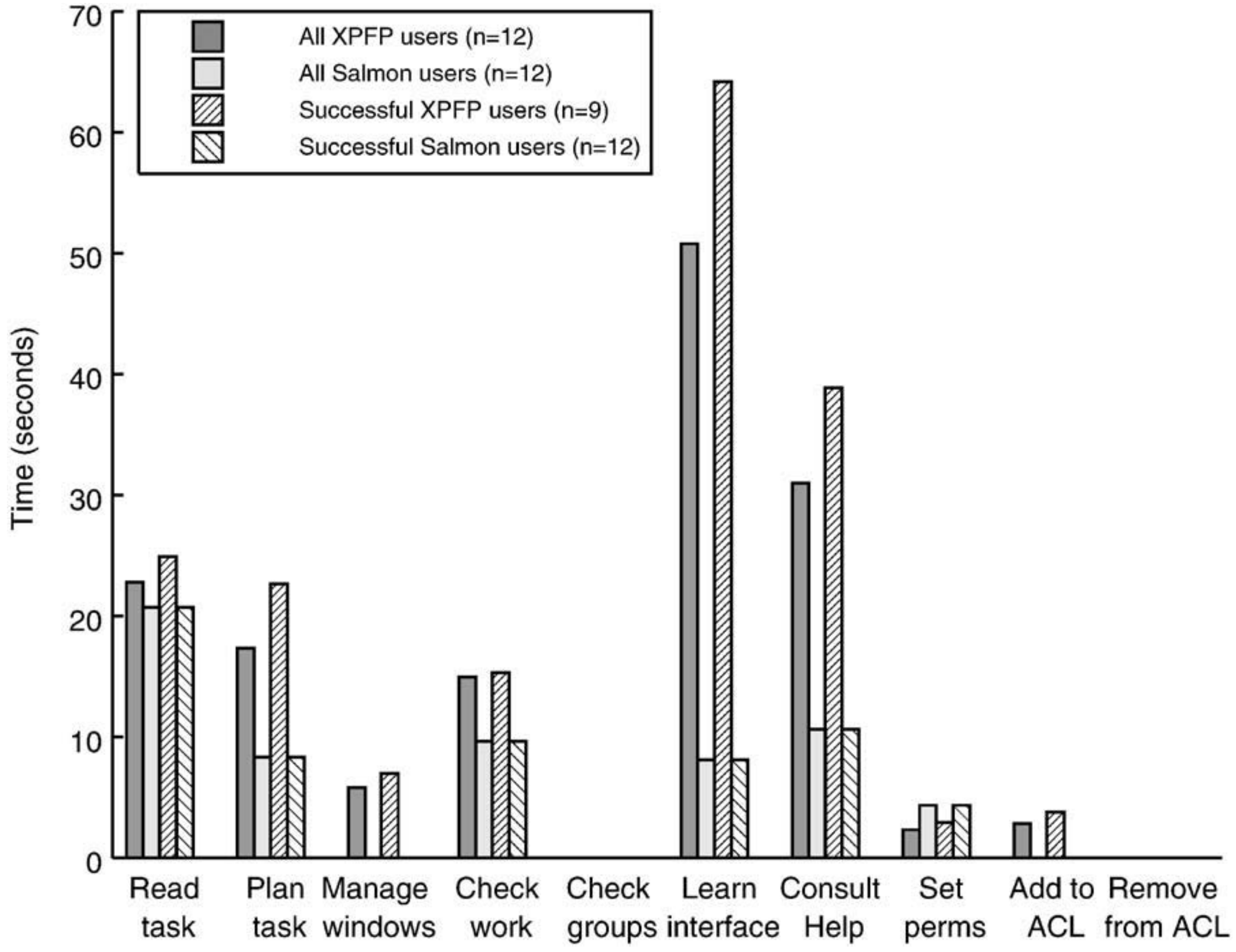


Figure 3.6: *Average time spent on each of 10 behaviors identified in the Tux task by all XPFP and Salmon users, whether successful or not, and all successful XPFP and Salmon users. Salmon users spent less time on information-gathering behaviors like plan-task, learn-interface, and consult-help.*

does not appear to have been the case; apparently confusion led many XPFP users to experiment more with that interface. This phenomenon was especially apparent in the Tux task, where XPFP users spent, on average, six times longer than Salmon users exploring and experimenting with the interface.

- *Consult Help*: This behavior consists of browsing or reading the Help files. Users consult Help when confused about what to do next or about how an interface works. XPFP users spent far more time consulting Help than did Salmon users, presumably because they were having more difficulty using the XPFP interface.
- *Set permissions*: This behavior consists of reading permissions labels and clicking on their corresponding checkboxes. In the Wesley task, after users had added Wesley to the ACL, they would scan through the list of permissions labels in the interface. Salmon lists all 13 atomic permissions labels, while XPFP lists only 6 composite permissions on its main window. Since there are fewer labels to scan in XPFP, those users set permissions faster, on average. However, the added efficiency in XPFP comes at a price; all of the users who failed at the Jack task failed in part because they did not notice that Jack had ADMINISTRATE permission. They failed because the ADMINISTRATE permission is hidden two screens away. Furthermore, in the Tux task, XPFP users spent far more time searching for the DELETE permission, since it is hidden in XPFP, but readily visible in Salmon.
- *Add to ACL*: This behavior consists of selecting a user or group, and adding it to the ACL. Salmon uses essentially the same dialog as XPFP does for adding users to the ACL, so little difference in the behavior times between the two interfaces is expected here.
- *Remove from ACL*: This behavior consists of selecting a user or group, and removing it from the ACL. The Remove operation was essentially the same in both interfaces, and was rarely used.

As this detailed behavior analysis shows, Salmon users spent less time than XPFP users on the behaviors related to information gathering (Check groups, Learn interface, and Consult Help) and spent a greater proportion of their time on essential task behaviors (Add to ACL, Set permissions, and Check work). This observation supports the hypothesis that an improved external representation of task-relevant information—due to the use of ESS in the design of Salmon—leads users to spend less time looking for information and leads users to find more readily the information they need to complete tasks.

3.9 Conclusion

In the course of completing tasks with a user interface, users look for information to formulate goals and to check progress. When the necessary information is misleading or absent, users fail to establish the correct goals and hence make goal errors. Goal errors may lead to total task failure, and thus impinge on interface dependability. Many goal errors can be prevented by providing an accurate, clear and salient external representation of the information needed to achieve the user's primary goal. A design method for producing such a representation has been named external subgoal support (ESS).

The Windows XP file permissions interface, which does not use ESS, was shown to have unacceptably low success rates—58%, 25% and 75%—on three representative file permission-setting tasks. Salmon, an alternative interface designed in accordance with the ESS principle, was shown to increase percentage of successes to 83%, 100% and 100% on the same tasks. Furthermore, user tests with Salmon showed a dramatic reduction in the occurrence of goal errors compared to XPFP, with 80% fewer goal errors on one task, 94% fewer goal errors on another, and 100% fewer on a third. In addition, Salmon maintained or substantially reduced time to task completion for all three tasks. These improvements in successful task completion, reductions in goal error occurrence, and time to task completion were due primarily to ESS. These speed and success rates far more closely approach what is needed for dependable user interfaces in mission-critical systems like those required for setting security-related configurations of servers, firewalls, personal workstations, etc.

CHAPTER 4

Usability Challenges in Security and Privacy Policy-Authoring Interfaces

This chapter is largely a reproduction of a paper co-authored with Clare-Marie Karat, John Karat, and Carolyn Brodie and published at INTERACT 2007 [104].

Despite the need for usable policy-authoring interfaces, numerous studies and incidents have shown that several widely-used policy-authoring interfaces are prone to serious errors. The "Memogate" scandal, in which staffers from one political party on the United States Senate Judiciary Committee stole confidential memos from an opposing party, was caused in part by an inexperienced system administrator's error using the Windows NT interface for setting file permissions [132]. Maxion and Reeder showed cases in which users of the Windows XP file permissions interface made errors that exposed files to unauthorized access [85]. Good and Krekelberg showed that users unwittingly shared confidential personal files due to usability problems with the KaZaA peer-to-peer file-sharing application's interface for specifying shared files [55]. This evidence suggests that designing a usable policy-authoring interface is not trivial, and that designers could benefit from a list of potential vulnerabilities of which to be aware.

This chapter reports the results of a user study which had the goal of identifying common usability challenges that all policy-authoring interface designers must address. The study employed SPARCLE [68], an application designed to support enterprise privacy policy authoring. However, the study was not intended to evaluate SPARCLE itself, but rather to reveal challenges that must be addressed in policy authoring in general. SPARCLE-specific usability issues aside, the study revealed five general usability challenges that any policy-authoring system must confront if it is to be usable:

1. **Supporting object grouping;**
2. **Enforcing consistent terminology;**
3. **Making default rules clear;**
4. **Communicating and enforcing rule structure;**
5. **Preventing rule conflicts.**

These challenges are explained in detail in the discussion in Sect. 4.5. Although these challenges have been identified from a user study in just one policy-authoring domain, namely enterprise privacy policies, a review of related work, presented in Sect. 4.6, confirms that the challenges identified here have been encountered in a variety of other policy-authoring domains, including file access control, firewalls, website privacy, and pervasive computing. This work, however, is the first we are aware of to describe policy-authoring applications as a general class and present usability challenges that are common to all.

4.1 Policy Authoring Defined

"Policy" can mean many things in different contexts, so it is important to give a definition that is germane to the present work on security and privacy policies. For the purposes of this work, a policy is defined as a function that maps sets of elements (tuples) onto a discrete set of results, typically the set {ALLOW, DENY} (however, other result sets are possible; for example, Lederer et al. describe a policy-based privacy system in which tuples representing requests for a person's location are mapped to the set {PRECISE, APPROXIMATE, VAGUE, UNDISCLOSED} [78]). Elements are defined as the attributes relevant to a specific policy domain; the values those attributes can take on are referred to as element values. Element values may be literal values, such as the username "jsmith," or they may be expressions, such as "if the customer has opted in." Policies are expressed through rules, which are statements of specific mappings of tuples to results. Policy authoring is the task of specifying the element values in a domain, specifying rules involving those elements, and verifying that the policy comprised by those rules matches the policy that is intended.

In the privacy policy domain, for example, a policy maps tuples of the form (<user category>, <action>, <data category>, <purpose>, <condition>) to the set {ALLOW, DENY} [10]. Here, user category, action, data category, purpose, and condition are elements. ALLOW and DENY are results. An example privacy policy rule would map the tuple ("Marketing Reps," "use," "customer address," "mailing advertisements," "the customer has opted in") to the value ALLOW, indicating that marketing representatives can use the customer address data field for the purpose of mailing advertisements if the customer has opted in. Here, "Marketing Reps," "use," "customer address," "mailing advertisements," and "the customer has opted in" are element values.

To take another example, in the file permissions domain, a policy maps tuples of the form (<principal>, <action>, <file>) to the set ALLOW, DENY. An example rule might map ("jsmith," "execute," "calculator.exe") to the value DENY, indicating that the user jsmith cannot execute the program calculator.exe.

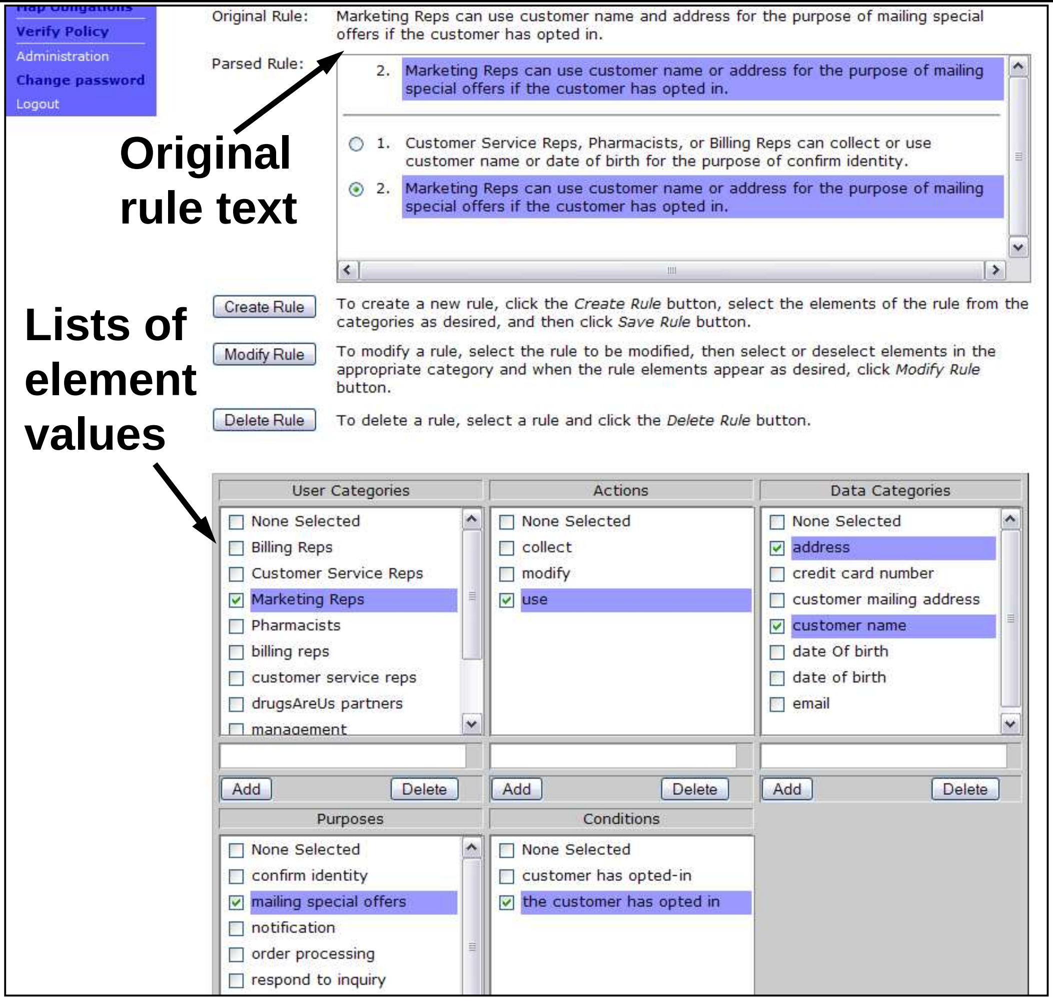
Since the rules in a policy may not cover all possible tuples, policy-based security and privacy systems typically have a default rule. For example, the SPARCLE system has the default rule that all 5-tuples of (<user category>, <action>, <data category>, <purpose>, <condition>) map to DENY. Additional rules are specified by policy authors and all author-specified rules map a 5-tuple to ALLOW. The default rule need not necessarily be a default DENY; a default ALLOW is also possible (policies with a default ALLOW rule are often called "blacklists," because any element value listed explicitly in the policy is denied access), or the default can vary according to some values in the tuples (for instance, a default rule might state that all accesses to shared files on a computer are allowed by default to local users but denied by default to remote users). Similarly, user-specified rules need not necessarily map exclusively to ALLOW or exclusively to DENY; it is possible to allow users to specify rules that map to either ALLOW or DENY. However, policy systems that allow users to specify both types of rules introduce the potential for rule conflicts, which can be a significant source of user difficulty [6, 29, 85].

4.2 The SPARCLE Policy Workbench

In the present work, policy authoring usability was investigated through a user study in which participants used the SPARCLE Policy Workbench application. SPARCLE is a Web-based application for enterprise privacy policy management. The application includes a front-end user interface for authoring privacy policies using a combination of natural language and structured lists. While the current embodiment of SPARCLE is tailored for the privacy policy domain, the user interface was designed with an eye toward supporting policy authoring in other domains such as file access control. From a research perspective, the two interaction paradigms supported by SPARCLE, natural language and structured list entry, can as easily be applied to privacy policy management as to file, network, system, email, or other policy management. Thus SPARCLE is a suitable tool for studying policy authoring in the general sense. Portions of the SPARCLE user interface relevant to the present study are described below; a more complete description of the SPARCLE system can be found elsewhere [69].

One way to write policy rules in SPARCLE is to use the natural language interface on the Natural Language Authoring page. The Natural Language Authoring page contains a rule guide, which is a template that reads, "[User Category(ies)] can [Action(s)] [Data Category(ies)] for the purpose(s) of [Purpose(s)] if [(optional) Condition(s)]." This template indicates what elements are expected in a valid rule. The Natural Language Authoring page further contains a large textbox for entering rule text. Policy authors can type rules, in natural language, directly into the textbox. For example, a rule might read, "Customer Service Reps, Pharmacists, and Billing Reps can collect and use customer name and date of birth to confirm identity." SPARCLE has functionality for parsing an author's rules so that it can extract rule element values automatically from the author's text. When parsing has completed, the user proceeds to the Structured Authoring page to see the structured format of the policy.

The Structured Authoring page, shown in Fig. 4.1, shows the results of parsing each policy rule. When SPARCLE parses each policy rule, it saves the elements (i.e., user categories, actions, data categories, purposes, and conditions) found in that rule. The elements are reconstructed into sentences and shown next to radio buttons in a list of rules. The lower half of the Structured Authoring page contains lists of element values, one list for each of the five elements of a privacy policy rule. These lists contain some pre-defined, common element values (e.g., "Billing Reps" as a user category, "collect" as an action, or "address" as a data category) as well as all element values defined by the policy author and found by the parser in rules written on the Natural Language Authoring page. Policy authors can also alter these element value lists directly by adding or deleting elements. When a rule is selected from the list of rules at the top of the Structured Authoring page, all of the element values in that rule are highlighted in the lists of element values. Policy authors can edit rules by selecting different element values from the lists. It is also possible to create rules from scratch on the Structured Authoring page.



Original Rule: Marketing Reps can use customer name and address for the purpose of mailing special offers if the customer has opted in.

Parsed Rule: 2. Marketing Reps can use customer name or address for the purpose of mailing special offers if the customer has opted in.

( ) 1. Customer Service Reps, Pharmacists, or Billing Reps can collect or use customer name or date of birth for the purpose of confirm identity.

(●) 2. Marketing Reps can use customer name or address for the purpose of mailing special offers if the customer has opted in.

To create a new rule, click the *Create Rule* button, select the elements of the rule from the categories as desired, and then click *Save Rule* button.

To modify a rule, select the rule to be modified, then select or deselect elements in the appropriate category and when the rule elements appear as desired, click *Modify Rule* button.

To delete a rule, select a rule and click the *Delete Rule* button.

User Categories	Actions	Data Categories
☐ None Selected	☐ None Selected	☐ None Selected
☐ Billing Reps	☐ collect	✓ address
☐ Customer Service Reps	☐ modify	☐ credit card number
✓ Marketing Reps	✓ use	☐ customer mailing address
☐ Pharmacists		✓ customer name
☐ billing reps		☐ date Of birth
☐ customer service reps		☐ date of birth
☐ drugsAreUs partners		☐ email
☐ management		

Purposes	Conditions
☐ None Selected	☐ None Selected
☐ confirm identity	☐ customer has opted-in
✓ mailing special offers	✓ the customer has opted in
☐ notification	
☐ order processing	
☐ respond to inquiry	

Figure 4.1: *SPARCLE's Structured Authoring page. Policy authors can create or edit rules on this page by selecting from the lists of element values in the lower half of the page.*

4.3 Policy Authoring Usability Evaluation

We conducted a laboratory user study using the SPARCLE application to identify usability problems experienced by users in policy-authoring tasks.

4.3.1 User Study Method

Participants

We recruited twelve participants, consisting of research staff and summer interns at a corporate research facility, for the user study. Participants varied in age from 20 to 49; four were female. Since participants did not have experience authoring privacy policies or using SPARCLE, we considered them novice users for our purposes. Participants were compensated for their participation with a certificate for a free lunch.

Table 4.1: *The task statement given to participants for the DrugsAreUs task, one of three tasks used in the user study.*

The Privacy Policy for DrugsAreUs
Our business goals are to answer customer questions when they call in (Customer Service), fulfill orders for prescriptions while protecting against drug interactions (Pharmacists), and to provide customers valuable information about special offers (Marketing). In order to make sure our customers' privacy is protected, we make the following promises concerning the privacy of information we collect at DrugsAreUs. We will only collect information necessary to provide quality service. We will ask the customers to provide us with full name, permanent address and contact information such as telephone numbers and email addresses, and a variety of demographic and personal information such as date of birth, gender, marital status, social security number, and current medications taken. On occasions where we need to verify a customer's identity, Customer Service Reps will only use the social security number to do so. Our pharmacists will use the current medication information when processing new orders to check for drug interactions.

We will make reports for our internal use that include age and gender breakdowns for specific drug prescriptions, but will not include other identifying information in the reports and will delete them after five years. For example, our research department might access customer data to produce reports of particular drug use by various demographic groups.

Apparatus

Participants accessed SPARCLE through the Internet Explorer web browser on a computer running Windows XP. SPARCLE ran on a server on a local intranet, so users experienced no network delays. We set up a camera and voice recorder in the laboratory to record participants' actions and words.

Training Materials

We presented participants with a 4-page paper tutorial on how to use SPARCLE to give them a basic introduction to the SPARCLE system. The tutorial walked participants through the SPARCLE Natural Language Authoring and Structured Authoring pages as it had participants write and edit a two-rule policy. The tutorial took about 15 minutes to complete.

Tasks

We wrote three task scenarios: the "DrugsAreUs" task, the "Bureau of the Public Debt" task, and the "First Finance" task. The three scenarios describe medical, government, and finance enterprises, respectively, and thus cover a broad range of the types of enterprises that require privacy policies in the real world. Each task scenario described an enterprise and its privacy policy requirements in language that did not explicitly state rules to be written into SPARCLE, but suggested content that might go into explicit rules. The intent of the scenarios was to give participants an idea of what to do, but to have them come up with their own language for their rules. An example of one of the three task scenarios, the "DrugsAreUs" task, is listed in Table 4.1.

Procedure

We asked participants to complete a demographic survey before beginning the user study. We then gave participants the SPARCLE tutorial and asked them to complete it. We provided help as needed as the participants worked through the tutorial. After they had completed the tutorial, we instructed participants to think aloud during the study [44]. We then presented participants with tasks. Each participant was presented with two of the three task scenarios and asked to complete them one at a time. We instructed participants to imagine they were the Chief Privacy Officer of the enterprise described in each scenario and to use SPARCLE to author the rules they thought were necessary to protect personal information held by the enterprise while still allowing the enterprise to carry out its business. We counter-balanced the selection and presentation of the scenarios across participants so that each scenario was presented to eight participants and each scenario was the first scenario presented to four participants and the second scenario presented to four other participants.

Data Collection

The data we collected included text of rules written, video of participants and the computer screen on which they worked, think-aloud audio, and results of the demographic survey.

4.3.2 Data Analysis Method

We performed two data analyses. In the first analysis, we looked at the rules participants wrote to find errors in rules. In the second analysis, we reviewed videos and think-aloud audio data to find any additional incidents of errors and usability problems not found in the first analysis.

First Analysis: Errors in Rules

In the first analysis, we read through participants' final rules and considered their implementability. An implementable privacy rule was defined as a rule that can be unambiguously interpreted by an implementer, which is a human or machine that produces the actionable code to carry out automated enforcement of a policy's intentions. With respect to implementability, we identified seven errors. Two of these errors, undetected parser errors and unsaved rule changes, are system-specific issues to SPARCLE, and are not further discussed here, because the objective of this work was to identify errors that might occur in any interface for policy-authoring. The five non-system-specific errors we found were group ambiguities, terminology mismatches, negative rules, missing elements, and rule conflicts. We identified and counted occurrences of each type of error. The five errors, criteria used to identify each type of error, and examples of each error are below:

1. **Group ambiguity:** Composite terms, i.e., terms that represented a set of other terms, were used in the same policy as the terms they apparently represented. This was considered an error for implementation because it was often not clear exactly which terms were represented by the composite term. For example, one rule said, "DrugsAreUs can collect necessary information...," in which "necessary information" is a composite term presumably representing data referred to in other rules such as "customer mailing address," and "current medications taken." However, it is not immediately clear just what data is referred to by "necessary information." As another example, one rule contained both the terms "contact information" and "permanent address." This would imply that, contrary to common usage, "permanent address" is not part of "contact information." An implementer could easily be confused as to whether the term "contact information" used in a different rule included permanent address data or not.

2. **Terminology mismatch:** Multiple terms were used to refer to the same object within the same policy. Examples of terminology mismatches included "email address" and "email addres;" "Financial control" and "Finacial control;" "gender" and "gender information;" "properly reporting information to the IRS" and "providing required reports to the IRS."

3. **Negative rule:** A rule's action contained the word "not." Although SPARCLE is a default-deny policy system, implying that it is only necessary to specify what is allowed, some participants attempted to write negative rules, i.e., rules that prohibited access. These rules are unnecessary, and can lead to confusion on the part of an implementer who is expecting only positive rules. An example of a negative rule was, "Bureau of the Public Debt can not use persistent cookies...."

4. **Missing element:** A rule was missing a required element. The missing element was usually purpose. An example of a rule with no purpose was "Customer Service Reps can ask full name, permanent address, and medication taken."

5. **Rule conflict:** Two different rules applied to the same situation. Only one rule conflict was observed in our study. SPARCLE's policy semantics avoid most potential for rule conflict by taking the union of all access allowed by the rules except in the case of conditions, for which the intersection of all applicable conditions is taken. The one observed example of a rule conflict included the rules, "Customer Service Reps can access customer name for the purpose of contacting a customer **if the customer has submitted a request**," and "Customer Service Reps can access customer name for the purpose of contacting a customer **if the customer has expressed a concern**." Since the user category ("Customer Service Reps"), action ("access"), data category ("customer name"), and purpose ("contacting a customer") all match in these rules, taking the intersection of conditions would imply that "Customer Service Reps can access customer name for the purpose of contacting a customer" only when both "the customer has submitted a request" and "the customer has expressed a concern" are true. Thus, in the case that the customer has submitted a request but not expressed a concern, the first rule would seem to apply but is in conflict with the latter rule.

Second Analysis: Review of Video and Think-Aloud Data

In the second analysis, we reviewed video of user sessions and transcripts of user think-aloud data for critical incidents which indicated errors or other usability problems. We defined critical incidents in the video as incidents in which users created a rule with one of the errors indicated in the first analysis but subsequently corrected it. We defined critical incidents in the think-aloud data as incidents in which users expressed confusion, concern, or an interface-relevant suggestion. Once we had identified critical incidents, we classified them according to the error or usability problem that they indicated. While critical incidents were caused by a variety of SPARCLE-specific usability problems, only those incidents relevant to the five general policy-authoring rule errors identified in the first data analysis are reported here. There were no critical incidents that indicated general rule errors not already identified in the first data analysis; thus, the second data analysis simply served to confirm the results of the first through an independent data stream.

Below are some typical examples of user statements classified as critical incidents, followed by the error under which we classified them in parentheses:

- "I don't want to have to write out a long list of types of information without being able to find a variable that represents that information to be able to label that information. In this case the label might be personal information defined to include customer name, address, and phone number." (Group ambiguity)
- "I'm not sure how to do negations in this template." (Negative rule)

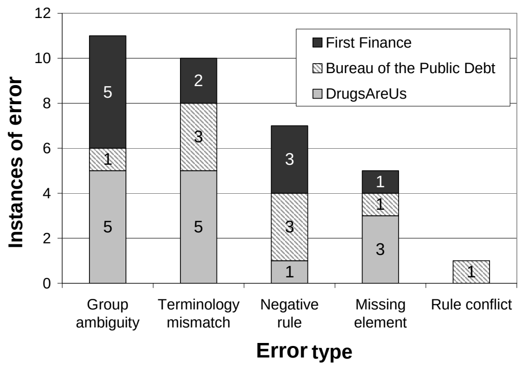


Figure 4.2: *Results of first and second data analyses, showing instances of five types of errors, broken down by task scenario in which they were observed. There were 24 total task-sessions, 8 sessions for each of the three tasks.*

- "It says I must specify at least one purpose, and I say, 'why do I have to specify at least one purpose?'" (Missing element)

4.4 Results

Results from the two data analyses, combined by adding the unique error instances found in the second analysis to those already found in the first analysis, are shown in Fig. 4.2. The errors in Fig. 4.2 are listed according to total frequency of occurrence, and within each error, are broken down by the task scenario in which they occurred. Since 2 of the 3 task scenarios were presented to each of 12 participants, there were 24 total task-sessions, 8 of each of the three scenarios. Thus, for example, the "group ambiguity" bar in Fig. 4.2 indicates that group ambiguity errors occurred in 11 of 24 task-sessions, including 5 of 8 "DrugsAreUs" sessions, 1 of 8 "Bureau of the Public Debt" sessions, and 5 of 8 "First Finance" sessions.

Of the five errors, group ambiguity errors were observed most frequently; a total of 11 instances of group ambiguity errors were found. The other errors, in order of frequency of occurrence, were terminology mismatches, negative rules, missing elements, and rule conflicts.

4.5 Discussion

The errors that participants made in this study suggest the five general policy-authoring usability challenges listed in the introduction to this paper. The challenges arise from inherent difficulties in the task of articulating policies; however, good interface design can help users overcome these difficulties. Group ambiguities suggest that users have a need for composite terms, but also need support to define these terms unambiguously. Terminology mismatches suggest the need for the interface to enforce, or at least provide some support for, consistent terminology. Negative rules are not necessary in SPARCLE's default-deny policy-based system, so users' attempts to write negative rules suggest that they did not know or did not understand the default rule. Missing elements are caused by users' failure to understand or remember the requirement for certain rule elements. Finally, rule conflicts are a known problem that a good interface can help address.

The identification of these five policy-authoring usability challenges is the primary result of this study. One of these challenges, communicating and enforcing rule structure, had already been anticipated, and SPARCLE's rule guide on the Natural Language Authoring page was designed to guide policy authors to write rules with correct structure. Rule conflicts, a well-known problem in policy-authoring domains, were also anticipated. SPARCLE, in fact, was designed to prevent rule conflicts by using a default-deny system and requiring that policy authors only write rules that mapped exclusively to ALLOW. It is only in the optional condition element that a conflict is possible in SPARCLE rules, so it was not surprising that only one rule conflict was observed in the present study. The remaining three usability challenges observed were largely unanticipated when SPARCLE was designed. Because of the fairly common failure to anticipate some of the five challenges, in SPARCLE and in other designs discussed above in the Introduction and below in the Related Work, the identification of these challenges is a significant contribution.

Before discussing the usability challenges observed, it is worth considering one methodological concern. Some of the errors revealed, particularly group ambiguities and negative rules, and the frequency with which they were observed in the present study, may have been influenced by the specific task scenarios presented to users. However, errors, except for the one instance of a rule conflict, are distributed fairly evenly across the three tasks, so it does not appear that any one task was responsible for eliciting a particular error type. Furthermore, the task scenarios were written based on experience with real enterprise policy authors and real enterprise scenarios, so the errors revealed are very likely to come up in the real world. However, the frequency values reported here should not be taken as necessarily indicative of the relative frequency of occurrence of these errors in real-world policy work.

Having identified five usability challenges for policy-authoring domains, it is worth discussing how these challenges might be addressed. Each challenge is discussed in turn below.

4.5.1 Supporting Object Grouping

Group ambiguities were caused by users not understanding what terms covered what other terms. Many solutions already exist to help users with tasks that involve grouped elements. Perhaps the most prominent grouping solution is the file system hierarchy browser. A hierarchy browser allows users to create groups, name groups, add objects to and remove objects from groups, and view group memberships. Hierarchy browsers may often be appropriate for policy-authoring tasks. However, hierarchical grouping may not always be sufficient. In file permissions, for instance, system users often belong to multiple, partially-overlapping groups. Any of a variety of means for visualizing sets or graphs may be useful here; also needed is a means for interacting with such a visualization to choose terms to go into policy rules. What visualizations and interaction techniques would best support grouping for policy authoring is an open problem.

4.5.2 Enforcing Consistent Terminology

Ambiguous terminology is nearly inevitable, but there are certainly ways to mitigate the most common causes, which, in this study, included typos and users' forgetting what term they had previously used to refer to a concept. A spell-checker could go a long way toward eliminating many typos, like "socail security number," in which "social security number" was obviously intended. A domain-specific thesaurus could help resolve abbreviations, aliases, and cases in which the same object necessarily has multiple names. For example, a thesaurus could indicate that “SSN” expands to “social security number,” and that “e-mail,” “email,” and “email address” all represent the same thing. A display of previously used terms for an object might help users remember to reuse the same term when referring to that object again; SPARCLE’s structured lists are an example of such a display. In some policy-authoring domains, the terminology problem may be resolved for the policy author by pre-defining terms. For example, in file permissions, the actions that can be performed on a file are typically pre-defined by the file system (e.g., read, write, and execute).

4.5.3 Making Default Rules Clear

Showing default rules may be a trivial matter of including the default rule in interface documentation or in the interface display itself. However, the concept of a default rule and why it exists may itself be confusing. One method of illustrating the default rule to users is to present a visualization showing what happens in unspecified cases [85]. SPARCLE includes such a visualization, although it is not onscreen at the same time a user is authoring rules [25].

4.5.4 Communicating and Enforcing Rule Structure

SPARCLE already does a fairly good job of enforcing rule structure. Participants in the present study recovered from forgotten purpose elements in 2 out of 5 cases due to SPARCLE’s prominent display of the phrase “None Selected” as the purpose element when no purpose was specified; a corresponding “Missing Purpose” error dialog also helped. Other interaction techniques like wizards, in which users are prompted for each element in turn, would likely get even higher rates of correct structure. Which of these techniques or combination of techniques leads to the fewest missed elements will be the subject of future work.

4.5.5 Preventing Rule Conflicts

Rule conflicts were rare in this study; the one observed conflict can be attributed to a lack of awareness about the semantics of the condition element. However, rule conflicts have been shown to be a serious usability problem in past work in other policy-authoring domains [6, 29, 85]. Clearly, interfaces need to make users aware of conflicts, and if possible, show them how conflicts can be resolved.

Rule conflicts have been the focus of some non-interface-related theoretical work [2, 50]; algorithms exist for detecting conflicts in a variety of policy contexts. This work could be leveraged by interface designers. However, the means for presenting policy conflicts to authors have yet to be evaluated. A few visualizations and interfaces have attempted to do this, such as those discussed below in the Related Work [6, 29, 85], but it is not clear whether they succeed at conveying conflicts, the need to resolve them, and the means to resolve them to users.

4.6 Related Work

Although the present study looked for usability challenges in just one policy-authoring domain, related work in other domains confirms that the usability challenges identified here are general problems encountered in a variety of policy-authoring domains. However, past work has only identified the challenges as unique to specific domains, rather than as part of the more general policy-authoring problem.

A need for supporting groups of element values has been found in several domains. Lederer et al. found a need for supporting groups of people in a user interface for setting location-disclosure policies in a pervasive computing environment so that policies could be scaled to environments with many people [76]. They present an interface for setting location-disclosure policies, but mention support for grouping as future work. The IBM P3P Policy Editor is a policy-authoring interface that uses hierarchical browsers to show users how Platform for Privacy Preferences (P3P) element values are grouped [40]. Zurko et al.'s Visual Policy Builder, an application for authoring access-control policies, allowed authors to create and label groups and to set constraints on groups to prevent conflicts due to group overlaps [153].

Finding good terminology and using it consistently has long been recognized as a problem for virtually every user interface [93]. In the policy-authoring area, Cranor et al. acknowledged the difficulty of finding comprehensible terminology in developing an interface for allowing users to specify what privacy practices they prefer in websites with which they interact [41]. Good and Krekelberg found that the use of four different terms for the same folder confused users in an interface for specifying shared folders in the KaZaA peer-to-peer file-sharing application [55].

Communicating default rules has been shown to be a problem in setting file permissions by both Maxion and Reeder [85] and Cao and Iverson[29]. Cranor et al. also discuss their efforts to come up with an appropriate default rule and communicate that rule to users [41]. Good and Krekelberg found that that KaZaA did not adequately communicate default shared files and folders to users [55].

The above-referenced independent studies of file-permissions-setting interfaces, Maxion and Reeder [85] and Cao and Iverson [29], also found that users have difficulty detecting, understanding, and correcting rule conflicts in access control systems. Zurko et al. considered the problem of conveying rule conflicts to users in their design of the Visual Policy Builder [153]. Al-Shaer and Hamed acknowledge the difficulties that rule conflicts cause for authors of firewall policies [6]. Besides human-computer interaction work, some theoretical work has acknowledged the problem of rule conflicts and found algorithms for detecting conflicts in policies [2, 50].

Lederer et al. report five design pitfalls of personal privacy policy-authoring applications that do not include the same usability challenges listed here, but do raise the important question of whether configuration-like policy-authoring interfaces are needed at all [75]. They argue that in personal privacy in pervasive computing environments, desired policies are so dependent on context, that users cannot or will not specify them in advance. While their argument is undoubtedly correct for some domains, there remains a need for up-front policy authoring in many situations: the system administrator setting up a default policy for users, the policy maker in an enterprise writing a privacy policy to govern how data will be handled within the organization, and even the end-user who does not want to be bothered with constant requests for access, but prefers to specify up-front what access is allowed.

4.7 Conclusion

In order to be usable, policy-authoring interfaces, which are needed for a wide variety of security and privacy applications, must address the five usability challenges identified in the user study described in this paper: supporting object grouping, enforcing consistent terminology, making default policy rules clear, communicating and enforcing rule structure, and preventing rule conflicts. Some of these issues have been addressed before in domain-specific policy-authoring interfaces and elsewhere, but all might benefit from novel, general interaction techniques. As more policy authoring interfaces for users are created to fit the variety of applications that depend on accurate policies, researchers and designers would benefit from considering the five usability challenges discussed in this paper and creating innovative interaction techniques to address them.

PART III

Expandable Grids

CHAPTER 5

Policy Authoring and the Expandable Grid Concept

In this chapter, we describe policy authoring, give definitions of its concepts, and list four fundamental policy-authoring operations that policy-authoring interfaces should support. We then describe the Expandable Grid concept as it applies to policy authoring in general. Finally, we list features of different policy domains and discuss which of these features make a policy domain suitable or not suitable for display in an Expandable Grid interface.

5.1 Definitions of policy-authoring concepts

The concepts surrounding policy authoring are perhaps most easily understood by example. We will use file access control as an example of a specific domain within the general domain of policy authoring, then we will explain how the concepts behind file access control policies can be generalized to other policy-authoring domains.

5.1.1 File access control as an example policy-authoring domain

In a basic file access control system, there are resources, principals, and actions. Resources are files and folders, the data to which access is to be controlled. Principals are system users, processes, computers, and other entities that may attempt to access resources. Actions are operations that may be performed on resources, such as "read," "write," "execute," and "delete." A *request* is an attempt by a principal to perform an action on a resource. A request can be represented as a set of principal, action, and resource. We call the components of a request, i.e., principal, action, and resource, *attributes*, and specific values for principal, action, and resource *values*. To give an example of these concepts, a request by a user named jsmith to execute a file called calculator.exe could be represented as the tuple ("jsmith," "execute," "calculator.exe"). The values would be "jsmith," "execute," and "calculator.exe."

A *policy* is a function that maps requests to *results*, which are drawn from the *result set* {ALLOW, DENY}, indicating whether the request is granted or not. We call a specific mapping of values to a result a *decision*. The actual decisions that will be issued by a policy are specified by *policy authors*, or simply *authors*, who are people authorized to control resources. Policy authors might be system administrators who control many resources for an organization or end users who may control files or personal information belonging to them that resides on organizational systems or in their own pervasive devices.

Policy authors often specify policies as a set of rules. A rule is a statement of a specific mapping of one or more requests (each consisting of a principal, action, and resource) to a result. For example, a rule might map the request ("jsmith," "execute," "calculator.exe") to the decision DENY, indicating that the user jsmith cannot execute the program calculator.exe. If a policy author put this rule into place, then if the request ("jsmith," "execute," "calculator.exe") were made, the access control system's decision would be to output DENY.

Because a given set of rules may not apply to all possible requests, a file access control system will have a *default rule* that indicates whether requests are allowed or denied by default. A typical default rule is default-deny, which states that all requests not explicitly granted by other rules are denied. A default-allow is also possible (spam or firewall "blacklists" are examples of default-allow policies) or the default rule can produce both ALLOW and DENY decisions depending on input values (for instance, a default rule might state that all accesses to shared files on a computer are allowed by default to local users but denied by default to remote users).

Since a file system may have thousands of principals and files, file access control systems typically provide a means for constructing *composite values*, which are collections of values. Values that are not composite are *atomic values*). For instance, in hierarchical file systems, files are naturally aggregated into folders and folders into higher-level folders. Principals might be placed into groups or roles according to their position in an organization, such as "Executives," "Managers," and "Employees." (The term "role" comes from role-based access control, in which a role is a construct for associating principals with actions and resources according to organizational functions [115].) Specific files and users are atomic values while folders and groups or roles are composite values. Groups and folders allow policy authors to write rules that apply to all components of a composite value, such as a rule that denies access to all employees to delete files in a folder called "Critical files." Composite values thus allow for more succinct policies, since a single rule can apply to many input values, such as any of multiple employees. Composite values also allow for policies that cover as-yet-unknown input values; the example rule above would apply to any future employees or future files added to the "Critical files" folder.

Use of composite values can lead to *rule conflicts*, which are situations in which more than one rule applies to a given request. To give an example of a rule conflict, suppose there is Rule 1 that states that the group "Employees" can read a file called "CompanyData." Now suppose there is a Rule 2, concerning an untrustworthy employee named "Jack," that states that Jack is not allowed to read the "CompanyData" file. If Jack tries to read the file, these two rules will both apply: Rule 1 because Jack is a member of the "Employees" group, and Rule 2 because it applies directly to Jack. To resolve rule conflicts, access control systems have *precedence rules*, which dictate which rules take precedence in case of a conflict. Several precedence rules are possible. A typical precedence rule is deny-takes-precedence, in which any rule mapping a request to DENY takes precedence. In the Jack example, Rule 2 would take precedence under a deny-takes-precedence rule. Another possible precedence rule is order-precedence, in which precedence goes to the rule the policy author has specified earliest. In the Jack example, Rule 1 would take precedence under order precedence.

It is possible for an access control system to avoid rule conflicts by only allowing policy authors to specify exceptions to the default rule, and not allowing exceptions to the exceptions. For example, if the default-deny rule is in effect and policy authors can only specify requests that should be allowed, no conflicts will arise (in fact, there would be "conflicts" with the default rule, but the conflicts would be resolved because author-specified rules always take precedence over the default rule). However, such a system can be unwieldy in certain situations. For example, if all members of a large group of principals except one is to be allowed access to a file (as in the Jack example), it would be necessary to form a new "all-but-the-exception" group or to specify a rule allowing access for every single member of the large group except for the one. Access control systems where situations such as these may arise typically allow policy authors to specify rules that either allow or deny access; hence, rule conflicts may arise in which author-specified rules conflict with each other.

5.1.2 Generalizations to other policy-authoring domains

The example of file access control illustrates all of the concepts behind policy-authoring domains in general. However, it should be noted that other policy-authoring domains may have different input dimensions or different result sets.

Enterprise privacy policies are a policy-authoring domain with more attributes than file access control. Requests in systems using enterprise privacy policies may have purposes, obligations, and conditions in addition to principals, actions, and resources [10]. Purposes are reasons for making a request, such as "sending marketing materials," obligations are requirements on data usage that must be completed after the request, such as "data must be deleted within one year," and conditions are miscellaneous properties that must hold prior to a request in order for the request to be granted, such as "it is before 5PM on a weekday." Several policy-authoring domains have time as an input dimension. For instance, Grey [18], a building-access system in which cell phones are used to open office doors, allows a user to give access to their office to another user for a limited duration (e.g., one week). In a future version of Grey, it might be possible to allow access to a door only between 9am and 5pm, for example.

It is possible to have a policy-based system in which requests can be partially or conditionally granted, in which case the result set may be more than just {ALLOW, DENY}. For example, Lederer et al. describe a policy-based privacy system in which requests for a person's location are mapped to the set {PRECISE, APPROXIMATE, VAGUE, UNDISCLOSED} [78]. As another example, APPEL, A P3P Preference Exchange Language, matches a user's privacy preferences to a website's practices and returns results from the set {REQUEST, LIMITED, BLOCK} (which are equivalent to allow, allow with conditions, and deny). APPEL can also return text strings to the user explaining its decisions.

5.1.3 Notes on the definitions of policy and policy authoring

One could define a policy as a set of rules. However, we have intentionally defined a policy as a function, and stated that a set of rules is merely a means for specifying a policy rather than the definition of a policy. We make this distinction because many of today's list-of-rules policy-authoring user interfaces imply that a policy is merely the sum of a set of rules, and fail to make users aware that because of default rules, composite values, rule conflicts, and precedence rules, a policy is actually more than the sum of the rules a user specifies. With our visualization-centered approach to policy-authoring interface design, a policy is presented to users as a function that maps requests to results, so users will see exactly what decisions their policies will result in, rather than have to reconstruct policies in their heads by taking their own rules, default rules, composite values, and precedence rules into account. We sometimes use the term *effective policy* to clearly distinguish our notion of policy as a function from the notion of a set of rules. Effective policy is the function that results from a set of rules after the application of defaults and the resolution of conflicts.

The description of file access control we give above is an adaptation of the access matrix model first described by Lampson [73]. The access matrix model is very general, so many policies can be represented in a matrix. Other well-known access-control models, such as the Bell-LaPadula [21], Chinese Wall [24], and role-based access control [115] models, can be thought of as implementations of the access matrix model, or at least can be represented by an access matrix [74].

5.2 Policy-authoring operations to support

While different policy-authoring domains may vary in their objective and in the specific mechanics of how their policies work, they have certain fundamental operations in common. Listing these fundamental operations is important in testing the ideas in this thesis, since by demonstrating that a visualization-centered user interface effectively supports a fundamental operation, we can be more confident that the technique will be effective in any policy domain that involves the operation, and thus make more general claims about when the visualization-centered approach is effective.

We have observed people authoring security and privacy policies in two laboratory studies we conducted [85, 104]. The fundamental policy-authoring operations we have identified are:

1. *Viewing policy decisions.* Policy-authoring interfaces should make it easy to look up what decision will result from a given request for access. Thus, they should show effective policy, which is the mapping of requests to decisions, rather than merely showing the rules that combine to make the effective policy. Furthermore, they should include the ability to check general properties the author would like the policy to have, such as “no one should be allowed to delete this file,” or “no remote users should have access to this folder.”

2. *Changing policy decisions.* Naturally, policy-authoring interfaces should allow for making rules, which alter the decisions a policy will make. In addition, a policy should be editable, since policy authoring is often done in response to some situation, such as access being denied to someone who needs it.

3. *Viewing composite value memberships.* Composite values (e.g., groups of users, folders of files, collections of actions, etc.) make it much easier to specify policy rules that apply to many individual values, such as “All of my friends are allowed to read files in my ‘Music’ folder.” When a policy author is setting a rule involving one or more composite values, it is important to know just what the composite values contain. A policy-authoring interface should show the individual members of composite values.

4. *Detecting and resolving conflicts.* Rule conflicts occur when two rules apply to a particular request for resources. This may happen, for example, if a user is explicitly denied access to a resource, but is a member of a group that is explicitly allowed to access that resource. While policy-based systems typically adopt some means of resolving such conflicts, some of our past work has shown that users

have difficulty understanding how conflicts are resolved [85]. A policy-authoring interface should show conflicts to authors and help them resolve them.

Some other operations, including adding or removing resources or users, creating or deleting groups, changing group memberships, and controlling who has the authority to read, write, edit, and delete policy rules are certainly relevant to policy-authoring tasks, but they are generally supported by interfaces outside the policy-based system, such as a file system browser, an administrative console, or a content-authoring application. Thus, we do not consider them fundamental to policy-authoring as we are defining it. However, in Chapter 7, we discuss how we added some functionality to the basic file permissions Expandable Grid interface for moving files, creating groups, and changing group memberships.

5.3 The Expandable Grid concept

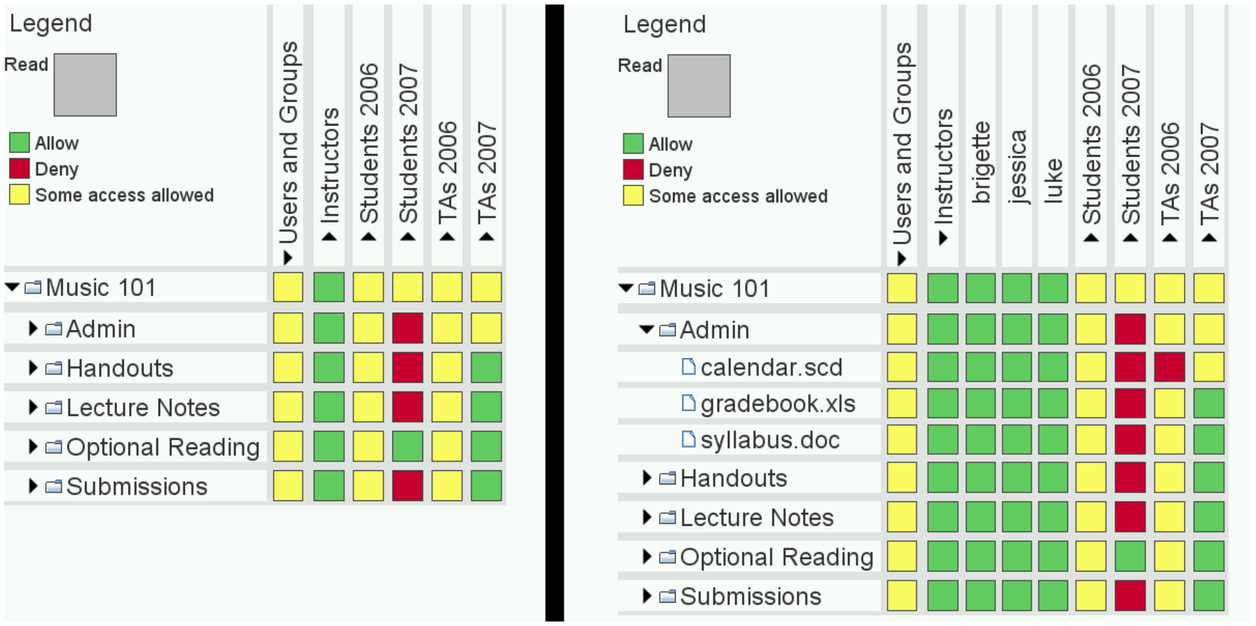
An Expandable Grid is a two-dimensional interactive matrix display. When displaying a policy in an Expandable Grid, two of the policy's attributes can be selected for display along the Expandable Grid's two axes. The attributes' values become the labels along the horizontal and vertical axes of the Expandable Grid. The labels define columns and rows that constitute a grid. At the intersection of each column and each row is a grid cell, which contains a graphical representation (such as a colored square) of the policy's decision for the values corresponding to the column and row labels. For an Expandable Grid to be truly expandable, and not just a static table, at least one of the two displayed attributes should have its values arranged in a hierarchy, which can be displayed along an Expandable Grid axis in an interactive tree, such as a Java Swing JTree component [84]. The interactive tree or trees along the axes of an Expandable Grid can be expanded to show more of a hierarchy's values or contracted to show fewer. When such expansion or contraction takes place, rows or columns are added to or removed from the grid, so that each visible hierarchy node always has a corresponding visible row or column (see Figure 5.1).

5.3.1 Benefits of the Expandable Grid

A policy-authoring interface based on an Expandable Grid offers several benefits over existing design paradigms. These benefits include visual pop-out effects to enable searches for anomalies in policy data, scalability for displaying a large policy in a small space, and a full-policy visualization that facilitates certain policy-authoring tasks. We discuss these benefits here.

5.3.1.1 Pop-out effects

"Visual pop-out" is a perceptual phenomenon in which a object in a visual search space is immediately distinguishable to a viewer from other objects in the space [130, 109, 136]. Pop-out may occur because an object's orientation, shape, color, motion, or other visual feature is different from most objects in the visual scene. In an Expandable Grid in which colored squares are used to represent policy decisions, visual pop-out makes it easy to spot anomalies. For example, if, in a file permissions policy, a particular group is meant to be denied access to all files in a particular folder, and red squares represent DENYdecisions in an Expandable Grid display of the policy, any green squares representing errant ALLOW decisions would pop out and be easily recognized by a policy author.



Legend

Read ▨

- 🟩 Allow
- 🟥 Deny
- 🟨 Some access allowed

	Users and Groups	Instructors	Students 2006	Students 2007	TAs 2006	TAs 2007
Music 101	🟨	🟩	🟨	🟨	🟨	🟨
Admin	🟨	🟩	🟨	🟥	🟨	🟨
Handouts	🟨	🟩	🟨	🟥	🟨	🟩
Lecture Notes	🟨	🟩	🟨	🟥	🟨	🟩
Optional Reading	🟨	🟩	🟨	🟩	🟨	🟩
Submissions	🟨	🟩	🟨	🟥	🟨	🟩

Legend

Read ▨

- 🟩 Allow
- 🟥 Deny
- 🟨 Some access allowed

	Users and Groups	Instructors	brigette	jessica	luke	Students 2006	Students 2007	TAs 2006	TAs 2007
Music 101	🟨	🟩	🟩	🟩	🟩	🟨	🟨	🟨	🟨
Admin	🟨	🟩	🟩	🟩	🟩	🟨	🟥	🟨	🟨
calendar.scd	🟨	🟩	🟩	🟩	🟩	🟨	🟥	🟥	🟨
gradebook.xls	🟨	🟩	🟩	🟩	🟩	🟨	🟥	🟨	🟩
syllabus.doc	🟨	🟩	🟩	🟩	🟩	🟨	🟥	🟨	🟩
Handouts	🟨	🟩	🟩	🟩	🟩	🟨	🟥	🟨	🟩
Lecture Notes	🟨	🟩	🟩	🟩	🟩	🟨	🟥	🟨	🟩
Optional Reading	🟨	🟩	🟩	🟩	🟩	🟨	🟩	🟨	🟩
Submissions	🟨	🟩	🟩	🟩	🟩	🟨	🟥	🟨	🟩

Figure 5.1: *Our file-permissions Expandable Grid interface before (left) and after (right) expanding the "Instructors" group node and the "Admin" folder node. When tree nodes are expanded, rows or columns are added to the grid so that each visible tree node has a corresponding row or column in the grid.*

5.3.1.2 Scalability

The expandability aspect of the Expandable Grid provides a degree of scalability for displaying large policies in a small space. Scaling to large datasets is one of the main challenges in the information visualization field [47], and a number of techniques have been designed for achieving scalability, such as distortion [80, 52, 116, 117], filtering [5, 49], zooming [20], searching, and aggregation [33, 63]. The Expandable Grid allows for scaling through aggregation; a grid cell that corresponds to a composite value on one or both axes displays a representation of the aggregate of the decisions for all the atomic values beneath the composite value. This aggregate representation may be all that a policy author needs to see, thus making it unnecessary to expand parts of the trees along the Grid axes and thus allowing the Grid to take up less visual space. For example, in Figure 5.1, the grid cell at the intersection of the Instructors group and the Admin folder is green, indicating that all decisions for members of the Instructors group are ALLOW for all files in the Admin folder. Thus, expanding the Instructors group and the Admin folder tree nodes is not necessary; the policy author already knows that all newly revealed grid cells will also be green, as shown in the figure.

5.3.1.3 Full policy visualization

The Expandable Grid shows the full policy in one view. The view may not all be visible on a display at once (our Expandable Grid implementations, for example, may require scrolling to see the entire policy), but it is readily available. Having the full policy in the same view enables policy authors to complete tasks that require searching large portions of the policy or making comparisons across parts of the policy. For example, finding all files accessible by a specific user is simply a matter of searching visually down a column in our file-permissions Expandable Grid. As another example, giving a new user the same permissions as some existing user is made easier by allowing both users' columns of policy data to be visible at the same time. Contrast the full policy view of the Grid with the file permissions list-of-rules approach, in which only a single rules or a few rules for a single resource can be viewed at the same time. Operations requiring searches of large portions of the policy or comparison across different parts of the policy are at least unwieldy, if not virtually impossible.

5.3.2 Problems common to Expandable Grid applications

The basic Expandable Grid must be adapted to a specific policy domain to make a fully functional policy-authoring interface, but some common problems may come up for many of the domains to which the Grid concept may be applied. We list those problems and suggested solutions here.

5.3.2.1 Mixed aggregate representation

In grid cells corresponding to composite values, an Expandable Grid can show a representation of an aggregate of the decisions of the atomic values beneath the composite values in the respective hierarchies. For example, a grid cell corresponding to a folder may show an aggregate of the decisions applying to all files in the folder. A common problem for the Expandable Grid interface designer is deciding what aggregate representation to use when the decisions applying to the atomic values are mixed. For example, in the file-permissions Expandable Grid, if decisions for all files beneath a folder are all ALLOW (or all DENY) (for a fixed principal and action), the natural aggregate representation for the folder's grid cell is green (or red)—the same as the ALLOW or DENY representation for a grid cell corresponding to atomic values. However, if the decisions are ALLOW for some files in a folder and DENY for others, it is unclear what representation should go in the folder's grid cell. In our file-permissions Expandable Grid, we used yellow squares to indicate such cases of "mixed decisions." A yellow square in a grid cell indicates that the policy author must expand nodes in one or both trees to see specifically what decisions apply to the atomic values beneath the composite values corresponding to the grid cell. In our P3P Expandable Grid, we used a color gradient to represent a mix of REQUIRED and NOT-USED decisions beneath a data category (the attribute shown along the vertical axis) and a black triangle (or "dog-ear") to indicate a mix of decisions beneath a purpose or recipient category (the attributes shown along the horizontal axis).

While the yellow squares, gradients, and dog-ears seemed to work adequately as mixed aggregate representations, we did observe some participants in our studies failing to expand the Grid interfaces, so the representations may not have been enough to suggest expansion to participants. We have not discovered better representations, however. Besides merely suggesting expansion, more could be done with mixed aggregate representations. For example, yellow squares in the file-permissions Grid could indicate the ratio of ALLOW decisions to DENY decisions they represent by their color saturation—light yellow might indicate a higher mix of ALLOW decisions, while a dark yellow might indicate a higher mix of DENY decisions.

5.3.2.2 Non-hierarchical attributes

The Expandable Grid was designed to show policies with attributes whose values are hierarchically structured, but it may be desirable to represent policy attributes whose values are not strictly hierarchical. For example, in file permissions, one user may belong to multiple groups. In our file-permissions Expandable Grid, we handle this issue simply by copying a username to every group to which the user belongs. See Figure 5.2.

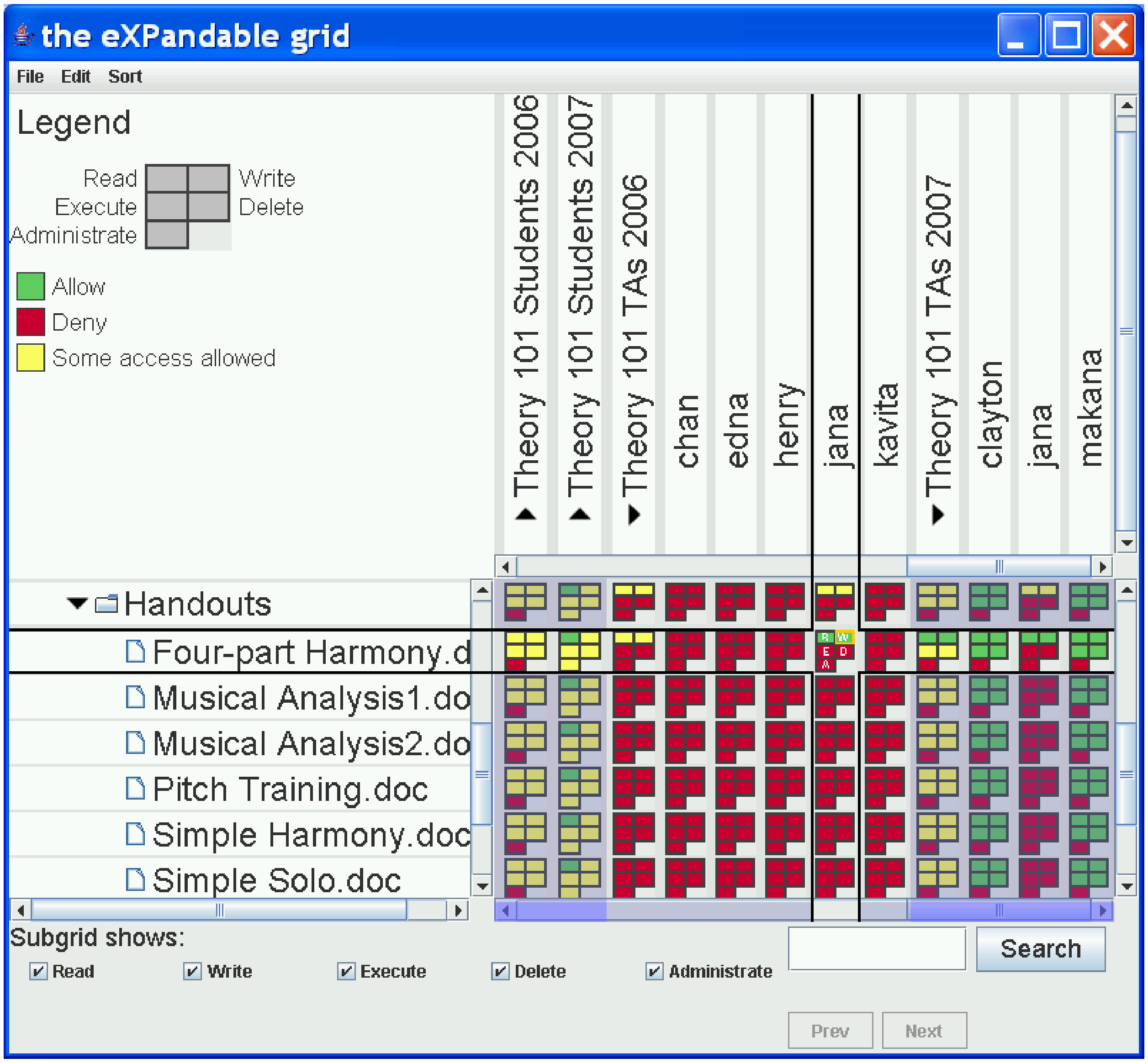


Figure 5.2: *A screenshot of our file-permissions Expandable Grid interface showing the user "jana" under two different groups.*

5.3.2.3 More than two attributes

The Expandable Grid can accommodate two policy attributes on each of its two axes, but when a policy has more than two attributes, other solutions must be devised. In our file-permissions Expandable Grid, the principal and resource attributes are displayed along the axes, but a third attribute, action, had to be displayed as well. Our solution was to use a *subgrid*, which is a set of small squares within the larger square of each grid cell. Each square of the subgrid represents an action—for example, for the five-square subgrids in the grid cells shown in Figure 5.2, the upper left square of each subgrid represents "read," the upper right square represents "write," the middle left square represents "execute," the middle right square represents "delete," and the lower square represents "administrate," as indicated in the interface's legend. The subgrids can create visual clutter, but our interface allows the policy author to select only a subset of actions to show in the subgrid. For example, Figure 5.3 shows subgrids of two squares that only show "read" and "write." The subgrid solution is not scalable; although we did not test its limits, it cannot be expected to show attributes with more than 10 or so values.

In our P3P Expandable Grid, we juxtaposed two attributes on the same axis; all recipients were shown on the left side of the horizontal axis, followed by all purposes on the right side. This juxtaposition of attributes did not work well, and we do not recommend it. The juxtaposition and its problems are discussed in detail in Chapter 8.

Although the P3P Expandable Grid's juxtaposition approach did not work, it is still possible to combine two attributes on the same axis by replicating the labels for one attribute under each of the labels for the other attribute. We tried this approach in an Expandable Grid for displaying Grey building access-control policies, shown in Figure 5.4. Principals and resources are combined along the horizontal axis in the Grey Expandable Grid interface. In the figure, notice the hierarchy of users, with the users "Prof Alice" and "Prof Bob" and groups "Students," "Visitors," and "Class" are copied under each of the resources "Cube," "Office," "Lab," (all under the expanded resource group "CUPS") "Kitchen," "Windows Computer," "Lounge," and "Classroom." We did not test the approach in user studies, so although it seems promising, we cannot make any claims about its usability.

Another approach to showing multiple attributes is to choose fixed values for some attributes and show only portions of the policy corresponding to those fixed values. This approach of setting fixed values for those dimensions not explicitly shown has been called slicing [63, 133]. A policy-authoring interface could provide a means (such as a dialog) for an author to choose fixed values for some attributes and could display the cross-section of the policy that remains after those attributes have been fixed. For example, in a privacy policy that has user category, action, data category, purpose, and condition attributes, the purpose could be fixed as "Marketing" and condition could be fixed as "if the customer has opted in," and an Expandable Grid interface could show the policy for all user categories (on one axis), data categories (on the other axis), and actions (in a subgrid) for those fixed values of purpose and condition.

Yet another approach is to simply show aggregate representations in grid cells, where aggregates are computed over all attributes that are not shown on the Expandable Grid axes [33, 63]. For example, in our file-permissions Expandable Grid, we could use this approach instead of using the subgrid to show actions. If a principal was allowed all five actions—"read," "write," "execute," "delete," and "administrate"—for a given resource, the corresponding grid cell would be green; it would be red if the principal were denied all actions to the resource; and it would be yellow if the principal were allowed some actions but not others. Combo boxes or other user interface widgets could be provided to allow the policy author to decide which attributes would actually be displayed along the axes.

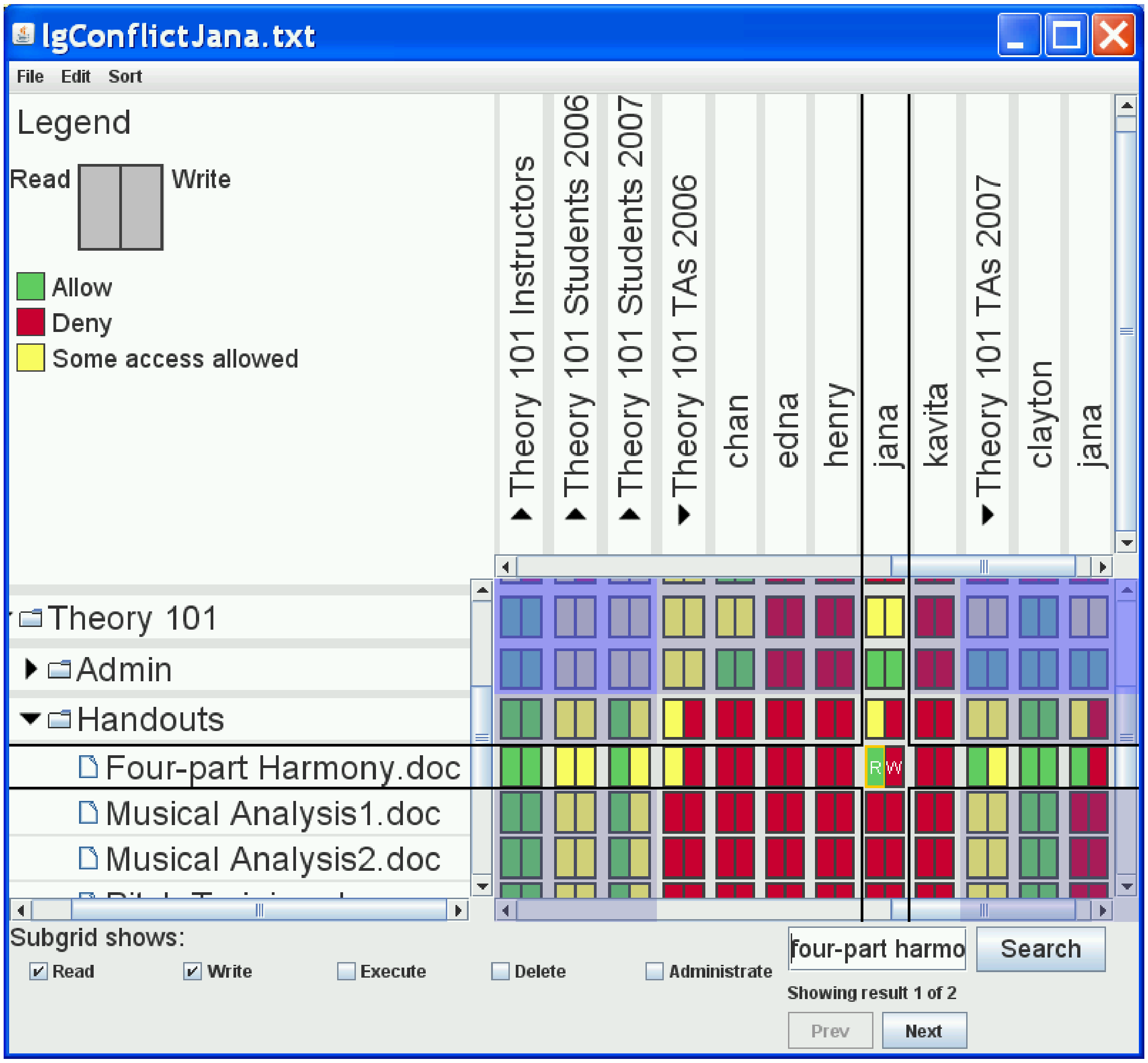


Figure 5.3: *A screenshot of our file-permissions Expandable Grid interface with two-square subgrids in each grid cell showing decisions for “read” and “write” actions.*

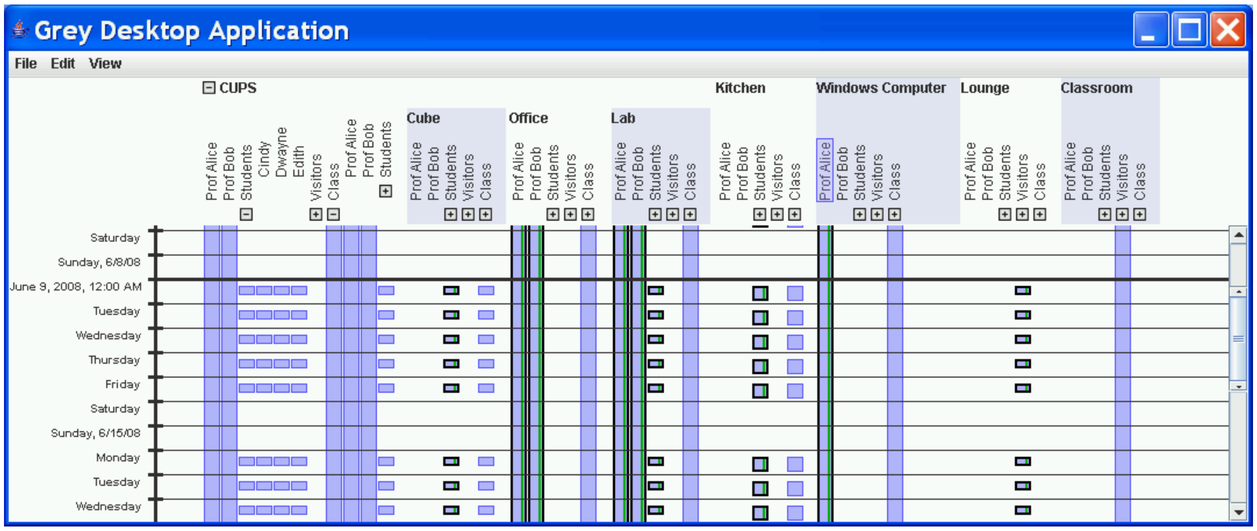


Figure 5.4: *A screenshot of our Grey Expandable Grid interface showing two policy attributes, resources and principals, combined on the horizontal axis.*

5.3.2.4 Vertical column labels

Rotated English text is harder to read than horizontally oriented text [28]. In our Expandable Grid designs, however, we have used vertically rotated labels for column headings. The vertically rotated text was, in fact, a usability problem, as discussed in Chapter 6. Moreover, we have seen people tilting their heads to read the column headings. Solving this problem may be harder than it first seems. Rotating text from vertical to 45 degrees may yield little readability improvement, and rotating the text any more toward the horizontal starts to make the column widths consume too much horizontal space. One possibility might be to rotate only those labels of the highest interest to full horizontal, while keeping labels in the periphery vertical. For example, we might rotate labels to horizontal when the mouse is hovering over them. Our research on this problem is ongoing.

5.3.2.5 Composite values as both containers and objects

Sometimes composite values in policies can serve both as containers of the values located beneath them in a hierarchy and as objects in their own right. For example, folders in a file system act as containers of files, but may also have their own permissions associated with them. This raises an issue for the Expandable Grid interface designer: how are we to show, at the folder level, both the policy associated with the files within the folder and the policy associated with the folder itself? We addressed this problem with a simple solution in our file permissions Expandable Grid interface: each folder contained a file called "." which is the common symbol for a folder self-reference. The row in the Grid corresponding to the "." file showed policy for the folder as an object, while the row corresponding to the folder itself showed policy for the objects the folder contained. See Figure 5.5. Unfortunately, participants in our user studies did not come across the "." in any of the tasks, so we do not yet know if it is an effective means for solving this problem.

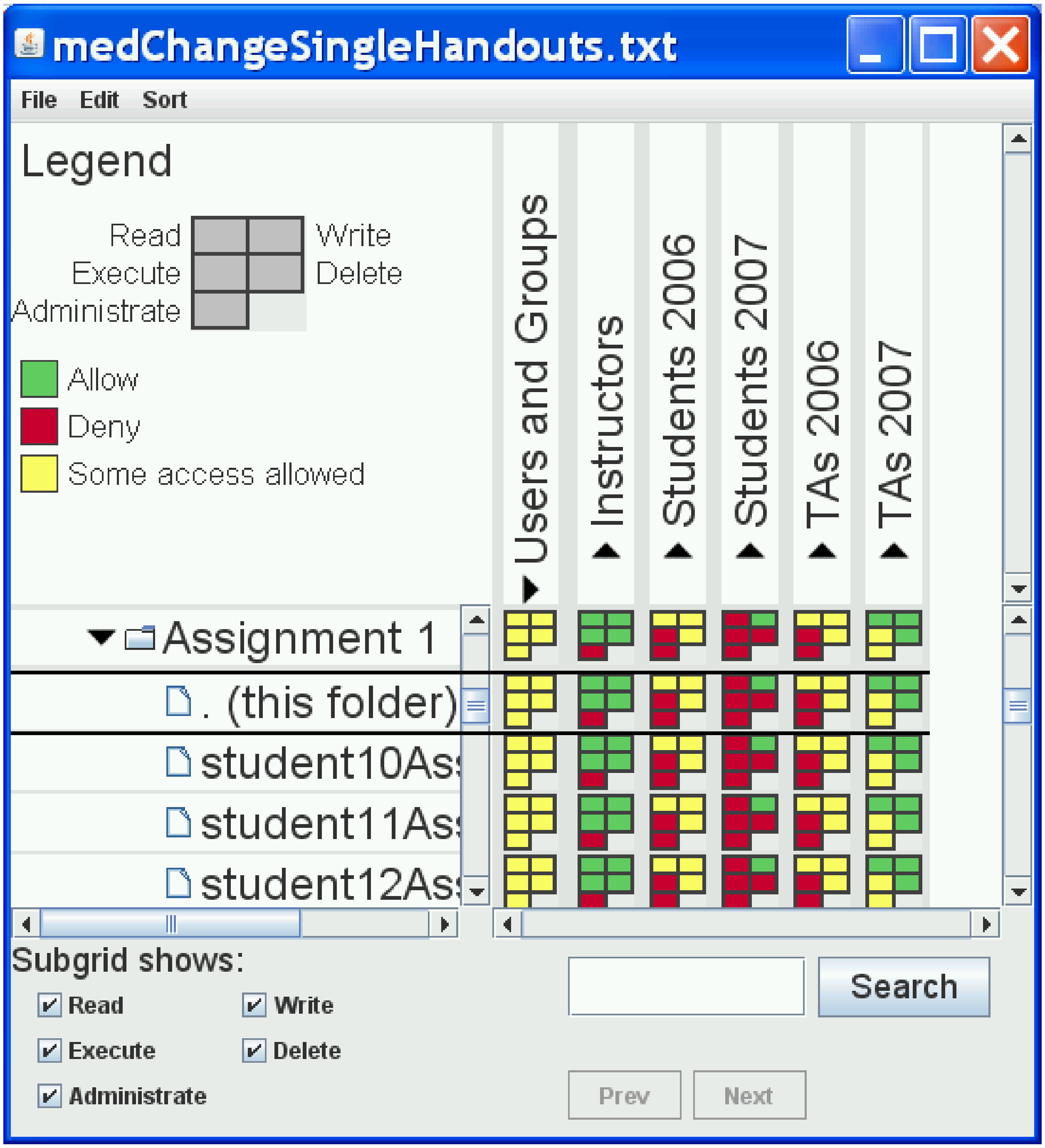


Figure 5.5: *Our file-permissions Expandable Grid interface uses the "." symbol to represent policy data for a folder as an object in its own right.*

5.3.2.6 Policy sandboxing

In general, it is undesirable for policy changes to go into effect immediately as an author makes them. It is better to allow an author to change policy in a safe "sandbox." With a sandboxing approach, the author can make a series of changes, see how they will affect the overall policy, decide that the policy is as it should be, and then put them into effect. Although our Expandable Grid interface designs have not addressed this issue, it could be addressed easily using the well-known "Apply" button idea, in which changes do not go into effect until the author clicks an "Apply" or "OK" button.

5.4 Applying the Expandable Grid concept

The fundamental ideas behind the Expandable Grid concept would seem, in theory, to apply to interface design for a wide range of policy domains. In practice, details of an Expandable Grid interface design need to be tailored to the specific domain, but with proper tailoring, the concept could be used in many domains. In this section, we discuss just what policy domains may be amenable to an Expandable-Grid-based policy display.

5.4.1 Characteristics of policy domains

We have already described in Section 5.1 our abstract concept of a policy, a function that maps attribute values to decisions. Here, we list some of the characteristics that vary amongst policy domains that may affect a domain's suitability for Expandable-Grid-based policy display:

- *Number of attributes.* The Expandable Grid is a 2-dimensional tabular display, so it is naturally suited to showing policies with two attributes. We have described how it can represent three attributes using a subgrid (if one of the attributes has only a handful of values). There are a number of techniques, described in Section 5.3.2.3, that could be used to adapt the Expandable Grid to show more than three attributes. However, the Grid in the embodiments shown in this thesis is best suited to displaying policies of no more than three attributes.

- *Label length.* A key feature of the Grid is that it summarizes a policy using graphical representations of policy data in the grid. This summarization may not be effective, however, if the labels of the rows and columns of the Grid are too long to both fit in a small space and to be comprehended at a glance. We discuss problems we ran into with long labels in Chapter 8.

- *Amount and importance of metadata.* Metadata is information about a policy that does not fit neatly into the Grid format. In designing our P3P Expandable Grid, we ran into problems finding a place to put metadata where it could be found when needed. Policies with a great deal of metadata may be best displayed in a format other than an Expandable Grid.

- *Type of decision.* As discussed in Section 5.1.2, policy decisions are not always binary. As the number of different decisions that have to be represented grows, an Expandable-Grid-based interface may grow more confusing. In our design of the P3P Expandable Grid, we ran into some problems with iconography—there were so many symbols to represent the four decisions of P3P plus aggregates of those decisions that study participants viewing the Grid became confused with the iconography.

- *Type of attributes.* The Expandable Grid was originally envisioned to display policies with hierarchical attributes, but not all attributes are hierarchical. Some attributes, such as action in file permissions or purpose in P3P, are flat categorical attributes, and some, such as time, are continuous. An Expandable Grid can display values for flat categorical attributes along its axes, but the flat structure will not take advantage of the Grid's scalability feature. As for continuous attributes, we have developed a prototype Expandable-Grid-based interface to show a continuous dimension along one of its axes. We present the prototype in Chapter 9.

5.4.2 Policy domains suitable for an Expandable Grid

Given our discussion of these policy domain characteristics, we can see which policy domains might be well-suited and which not well-suited to display in an Expandable Grid. Access-control policies are well-suited to display in an Expandable Grid: they have two or three attributes, short labels (usernames and file names), little metadata, binary decisions, and hierarchically structured attributes. Access-control policies, of course, are the focal application of the Expandable Grid in this thesis. Other policy domains that have similar characteristics and might be suitable Expandable-Grid-based policy displays include firewalls [6, 56, 145, 86, 15], enterprise privacy [26, 69, 104], location-disclosure [76, 110], and social networking [82].

5.4.3 Policy domains not suitable for an Expandable Grid

Other policy domains may not be well-suited to Expandable-Grid-based policy displays. Some characteristics that might make a policy domain ill-suited to Expandable-Grid displays include extensive natural language descriptions that are not easily summarized in grid form; long, flat categorical attributes; and very simple or very small policies for which a Grid would be overkill. Natural language website privacy policies without an accompanying P3P version would be one example of a policy domain ill-suited to display in an Expandable Grid. Examples of very small or simple policies include a coarse-grained location-sharing policy in which sharing is either "on" or "off," or a file system in which no files are shared except those placed in a particular "Public" folder. Such very small or very simple policies might be better summarized in a few sentences of text than displayed in an Expandable Grid.

CHAPTER 6

Expandable Grid Interface for Windows XP File Permissions

This chapter is largely a reproduction of a paper co-authored with Lujo Bauer, Lorrie Cranor, Mike Reiter, Kelli Bacon, Keisha How, and Heather Strong and published at CHI 2008 [103]. We thank Julie Downs, Joel Greenhouse, Bob Kraut, Howard Seltman, Steve Sheng, and an anonymous reviewer for help with our statistical analyses.

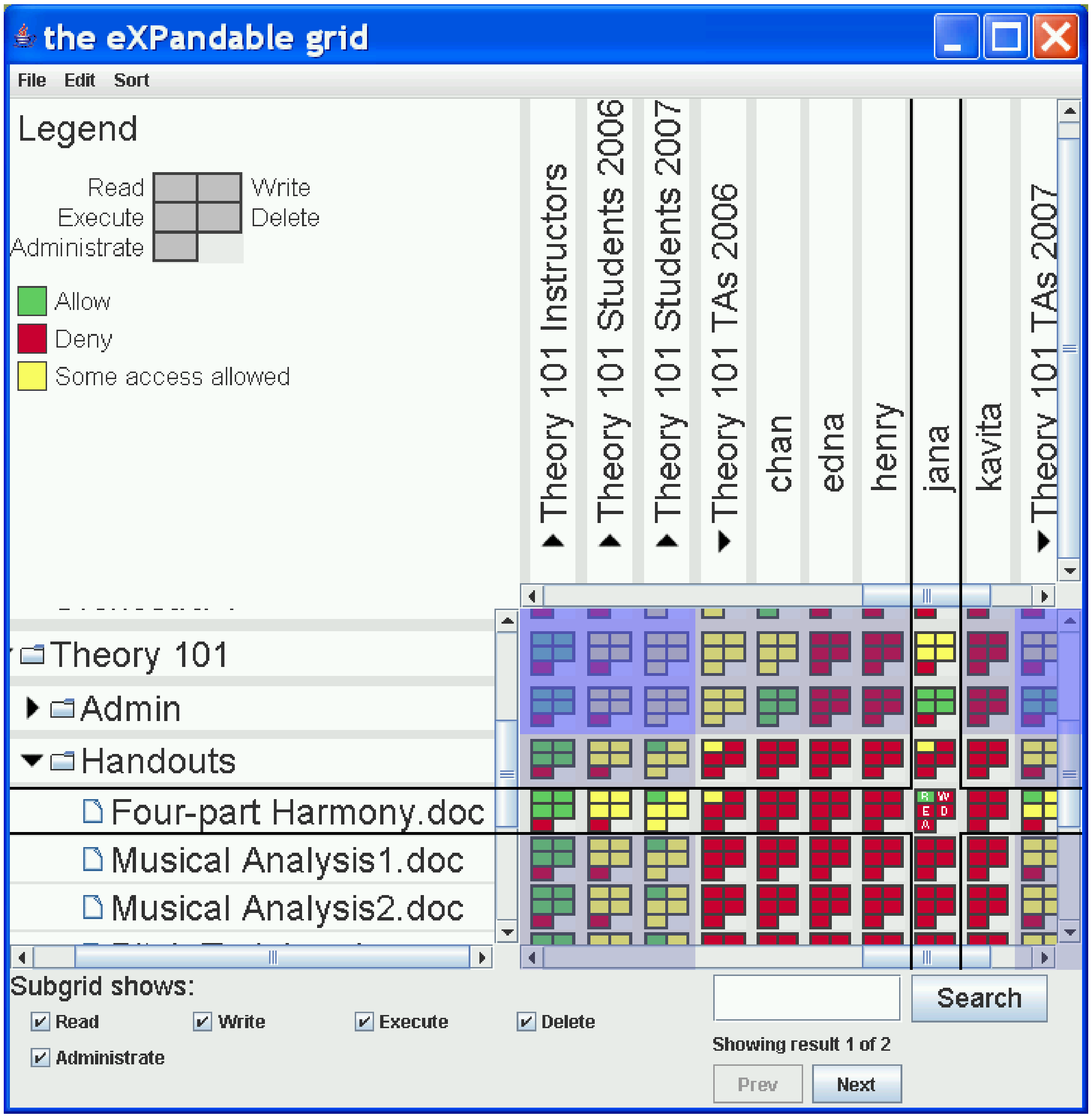
We implemented an interface based on the Expandable Grid concept for setting file permissions in the Windows NTFS file system. We ran a user study to compare authoring performance on a wide range of authoring tasks between our interface and the native Windows XP file-permissions interface. The Expandable Grid interface performed vastly better in our study; for example, the Expandable Grid interface achieved a 100% accuracy rate versus a 6% accuracy rate for the Windows interface for one task, and, on another task where accuracy rates were similar, achieved a 70% reduction in time-to-task-completion compared to Windows. We show specifically why the Expandable Grid interface performed better and conclude that it is likely a better option than the list-of-rules model for policy-authoring interface designs for file permissions and other security-related applications.

6.1 File-permissions Expandable Grid

We have already described the shortcomings of the list-of-rules model and particularly of the Windows XP file-permissions interface in Section 1.3.1, and we have introduced our solution, the file-permissions Expandable Grid in the same section. Here, we further discuss the features of the file-permissions Grid.

6.1.1 New policy semantics

One of the advantages of the file-permissions Grid is that when a policy author clicks on a green square, it turns red, and vice versa. Moreover, when policy is set at the group level, its effects are immediately propagated to the members of the group, so that the access indicated in the grid is always the access that will be given. For example, again referencing Figure 6.1, if the upper-right box in the grid cell at the intersection



Subgrid shows:

☑ Read ☑ Write ☑ Execute ☑ Delete

☑ Administrate

Showing result 1 of 2

Figure 6.1: *Screenshot of our Expandable Grid interface when the Jana task has been half-completed.*

of the group “Theory 101 TAs 2006” and the “Four-part Harmony.doc” file were clicked to change it from red to green, the upper-right boxes for all the group’s users—“chan,” “edna,” “henry,” “jana,” and “kavita”—would all turn green.

In order to ensure that clicking on the colored boxes in the Expandable Grid interface consistently has the same effect, we changed some of the semantics of Windows file permissions policies. Specifically, we changed the means by which rule conflicts are resolved. In Windows, when Jana is part of one group that is allowed access to a file and another that is denied access to that file, the deny rule takes precedence. If we were to retain the Windows semantics, clicking on a red box where a deny rule applies would leave the box red, rather than changing it to green as the policy author would expect. Even though clicking on the red box would add an allow rule, the box would stay red because the existing deny rule would still take precedence over the new allow rule. By our revised semantics, the most recently created rule takes precedence. Thus, if the box corresponding to Jana’s read access for the “Four-part Harmony.doc” file is red because of her membership in Theory 101 TAs 2006, but a policy author clicks on the box to make a rule that allows Jana access, this new rule will take precedence over the deny rule from the group and the box will turn green; Jana will now be allowed to access the file.

Our revised semantics makes for much easier resolution of rule conflicts than is possible in Windows. In Windows, resolving the Jana task requires either removing the deny permission not only for Jana, but also for all the other users in the Theory 101 TAs 2006 group, or removing Jana from the group entirely. Neither solution may be desirable, and in any case, policy authors find this dilemma confusing, if they even understand it at all [85].

Our semantics does have a potential drawback, however. The Imani task we designed for our study, which is described below, illustrates this drawback. Our description of the Imani task reads as follows:

> ....Two weeks ago, a scandal erupted in the Music Department... someone had been selling exam answers to the students! You suspected Imani, your fellow TA. Like the responsible administrator you are, you set file permissions so that Imani could not access the Answers folder under any circumstances.
>
> Now, it’s exam grading time, and your fellow TAs need access to the answers. Make sure they have access to the Answers folder, but be sure not to let Imani have access.
>
> Set permissions so that TAs 2007 (except for Imani) can read the Answers folder.

We set up the policy beforehand to have a deny rule applying to Imani for the Answers folder. In the Windows semantics, if a participant set a rule allowing the group TAs 2007 (which includes Imani) to read the Answers folder, Imani would still be denied access to the folder, even if the participant completely forgot to make an exception for Imani. In our semantics, if the participant set a rule allowing the group TAs 2007 to read the Answers folder, this rule would override the pre-existing rule denying Imani access, and, unless the participant went back and restored the rule denying Imani access, Imani would be allowed access.

6.1.2 Summary of Expandable Grid features

Our Expandable Grid interface has several features that we expect to provide advantages over the list-of-rules Windows XP file permissions interface. In particular, the Grid interface has these features:

- *Whole policy.* The Grid shows the whole policy, including principal/resource combinations for which there is no explicit rule.

- *Effective policy.* The Grid shows the effective policy, while Windows merely shows component rules.
- *Group membership information.* The Grid integrates group membership information into the file permissions display, while Windows puts it in a separate application from the file permissions interface.
- *Simple changes.* The Grid requires a simple click on a colored box to change a permission, while the Windows interface requires adding a new rule to its list.
- *New policy semantics.* The Grid's new policy semantics allows for easy conflict resolution by simply clicking on a colored box, the same way any other policy change would be made.
- *Visual pop-out.* The Grid allows for easy detection of anomalous permissions that visually pop out from the rest of the policy display.

6.1.3 Additional functionality

Our implementation of the Expandable Grid includes two additional functions worth mentioning: highlighting and search. Highlighting can be seen in Figure 6.1. The crosshairs follow the mouse pointer when it is positioned over the grid to help authors line up a grid cell with the principal and resource for which they wish to view or edit a rule. The search function can help users navigate a potentially large policy. Names of principals or resources can be typed into the search box. If the search term matches all or part of the name of a principal or resource, the principal or resource will be highlighted. If there are multiple search hits, the first will be highlighted, and other hits can be stepped through using the "Prev" and "Next" buttons beneath the search box.

6.2 User study method

We ran a laboratory user study with 36 participants to compare our Expandable Grid interface against the native Windows XP file permissions interface. We used a between-participants design, so 18 participants used the Grid and 18 used Windows; participants were randomly assigned to an interface condition. Each participant completed 20 tasks related to viewing or changing file permissions for a hypothetical Windows NTFS file server. The tasks are described in the "Task design" section below. We measured accuracy and time-to-task completion for each task.

6.2.1 Participants

We recruited 36 undergraduates in technical majors (science, engineering, and mathematics) to participate in the study. Ten were female. Participants were all daily computer users, but had never served as system administrators. Thus, our participant pool was consistent with our target demographic of novice or occasional access-control-policy authors, and was of a similar demographic as the system administrator of the Memogate scandal. While we believe it is important to make access-control interfaces usable for non-technical users as well as technical users, we leave testing the Grid on non-technical users as future work.

6.2.2 Experimental setup

Participants worked on a laptop running the Windows XP Professional operating system. Task statements were presented in a Web browser and were available on screen throughout the task. For participants using Windows, we started the Computer Management interface in a state that allowed for viewing system users and groups and opened Windows Help files related to setting file permissions. For participants using the Expandable Grid, the Grid application window was started at an initial size of 800x740 pixels. The laptop screen resolution was 1280x768.

The data we collected included the policies people created and screen video and audio of participants thinking aloud.

6.2.3 Task design

We designed a broad range of tasks to test each of the fundamental policy-authoring operations discussed in Section 5.2 in a variety of contexts and for both small- and large-scale policies. We created a scenario in which the participant was a teaching assistant (TA) in a music department, and was assigned to be the administrator of a file server that stored materials for classes in the department. We chose this scenario because it would be familiar to our participants while allowing for plausible tasks involving file permissions policies. The 20 tasks were designed around this scenario. Of these 20 tasks, 10 involved "small-scale" file permissions policies (involving roughly 50 principals and 50 resources) and 10 involved "large-scale" policies (roughly 500 principals and 500 resources). Each small-scale task had a large-scale parallel, so there were 10 pairs of tasks that varied in difficulty and in the fundamental operation they were testing. Also, each task pair tested one or more of the specific advantages or drawbacks we expected the Grid interface to have compared to the Windows interface. The following list describes what each of the ten task pairs required participants to do and which features (with reference to labels listed in the "Summary of Expandable Grid features" section above) of the Grid interface were tested by each:

1. *Training*: Make a simple policy change. The two tasks in this pair served to help the participant learn the interface they were using and get used to the scenario.
2. *View-simple*: Determine a user's access to a file when the user was inheriting access from a group. These tasks tested the Grid's *whole policy* and *group membership information* features.
3. *View-complex*: Determine a user's access to a file when a rule conflict is present. In this rule conflict, one rule applying to a specific user and one rule applying to a group of which that user is member were in conflict. These tasks tested the Grid's *effective policy* feature.
4. *Change-simple*: Change a principal's access to a resource. These tasks tested the Grid's *simple changes* feature.
5. *Change-complex*: Make multiple changes to a policy, including one involving a rule conflict. These tasks tested several of the Grid's features in one task, including the *whole policy*, *effective policy*, *group membership information*, *simple changes*, and *new policy semantics* features.
6. *Compare-groups*: Search for the intersection between two groups' members. These tasks tested the Grid's *group membership information* feature.
7. *Conflict-simple*: Detect and resolve a conflict in which the allow access a user is inheriting from a

Table 6.1: *Task statements given to participants for the small-scale conflict-complex and change-complex tasks, 2 of 20 tasks used in the user study.*

Conflict complex Nigel, another TA, has complained that when he was trying to edit the course calendar to change the due date for an assignment, he was denied access. Set permissions so that *Nigel* can *read and write* the *calendar.scd* file.
Change complex The gradebook file is highly sensitive. Instructors and TAs should be allowed to read it or write to it, but not delete it. And of course no students should be allowed to access the gradebook in any way. Make sure *instructors* and *TAs 2007* can *read* and *write* the *gradebook.xls* file. Make sure *instructors* and *TAs 2007* cannot *delete* the *gradebook.xls* file. Make sure *Students 2007* cannot access the *gradebook.xls* file in any way.

group should be overridden with an explicit deny rule. These tasks tested the Grid's *effective policy* and *simple changes* features.

8. *Conflict-complex*: Detect and resolve a conflict in which a user is allowed access through one group but denied access through another. The Jana task, discussed above, was the large-scale conflict-complex task. These tasks tested the Grid's *effective policy*, *simple changes*, and *new policy semantics* features.
9. *Memogate*: Detect that an unauthorized group has been given access to a resource. Our configuration for these tasks mimicked the vulnerability that made the Memogate scandal possible by including a policy rule that gave full access to all users for all files. These tasks tested the Grid's *visual pop-out* feature, because the unauthorized access causes the Grid to show a large area of green squares where red is expected, while Windows requires stepping through all rules to find the offending one.
10. *Precedence*: Allow a group to access a resource, but make an exception for one individual in the group. The purpose of these tasks was to demonstrate the drawback of our new policy semantics (the Imani task, discussed above, was the small-scale precedence task).

Examples of the text of task statements presented to participants can be seen in Table 6.1.

Some tasks, such as the tasks that involved viewing policy or group memberships, were multiple choice questions, so scoring was straightforward. The other tasks required making policy changes. For some of these tasks, there were multiple approaches to completing the task; in all cases, we scored a participant's final policy for a task according to whether they gave the access specified in the task statement; any extraneous policy settings were ignored.

6.2.4 Procedure

Our experimenter read instructions for our teaching-assistant scenario to participants. After reading these instructions, our experimenter read brief training materials that explained how to search for users and files, how to view permissions, and how to change permissions using the interface to which a participant was assigned. The experimenter also walked the participant through one full task as part of the training. For Windows participants, the training also covered how to use the Computer Management application and how to use the Windows Help files, since we considered those part of the file permissions interface; no help files were available to Grid participants. Training took about 3.5 minutes for Grid participants and about 5.5 minutes for Windows participants.

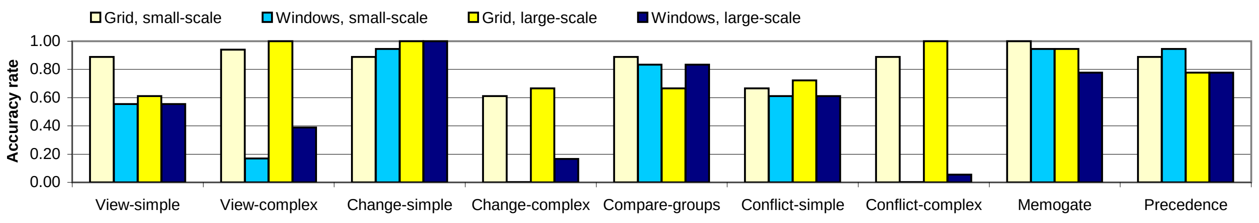


Figure 6.2: *Accuracy results, showing proportion of participants correctly completing each task with Grid and Windows interfaces.*

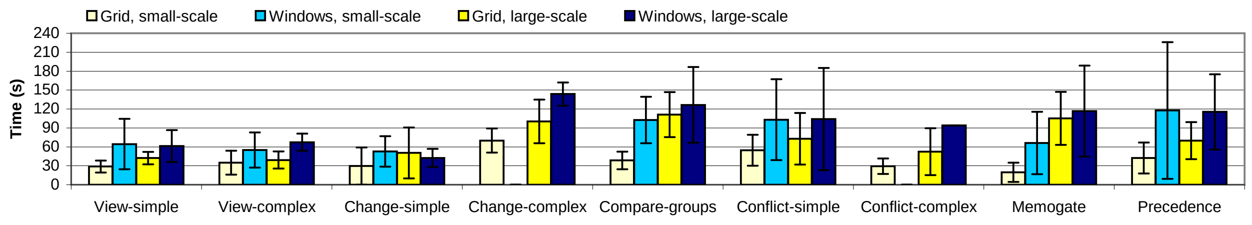


Figure 6.3: *Time-to-task-completion results, showing mean time-to-task-completion, in seconds, for participants who successfully completed each task with Grid and Windows interfaces. Error bars show +/- one standard deviation.*

Participants then began completing tasks. Before each task, the experimenter brought up the interface in a preconfigured state tailored to each task. Task statements were then presented to participants in a Web browser on screen. Participants were asked to think aloud while they worked on the tasks. Participants completed the 10 small-scale tasks first, then the 10 large-scale tasks. Each set of 10 tasks always started with the training task, but the order of the remaining 9 tasks were counterbalanced across participants using a Latin square design to guard against ordering effects.

6.3 Results

Our results are in the form of accuracy rates and mean time-to-task-completion for each interface condition and for each task. Accuracy rates represent the proportion of participants in each condition who correctly completed the task. The mean time-to-task-completion is computed only over the task completion times of those participants who successfully completed each task. Accuracy results can be seen in Figure 6.2. Timing results can be seen in Figure 6.3. Note that results are only shown for 18 tasks, 9 at each scale, because we excluded the two training tasks from analysis.

6.3.1 Experiment-wide results

For each metric, accuracy and time-to-task-completion, we performed an experiment-wide test of the hypothesis that the Grid interface performed better than the Windows interface over all tasks. We used logistic regression to test for a significant difference in accuracy scores between the Grid (overall accuracy 83.6%) and Windows (overall accuracy 56.5%) interfaces across all tasks. Logistic regression is a statistical technique used to test for effects of both numerical and categorical independent variables on a binary dependent variable [4]. In our case, interface, scale, and their interaction are our independent variables (also referred to as factors) and accuracy is our dependent variable (also referred to as a response). In using logistic regression, we fit a function to our data, just as we do in the more-familiar linear regression, but logistic regression uses the logistic function instead of a line. The logistic function is $f(\mathbf{x}) = \frac{1}{1+e^{-\beta_0 - \beta \mathbf{x}}}$, where β_0 is the intercept, $\mathbf{x}$ is a vector of the independent variables and β is a vector of coefficients. Logistic regression finds the values of β_0 and β that allow the logistic function to best fit the observed data. Precise interpretation of the β's can be complicated. For binary independent variables, like ours for interface and scale, the β's are the log-odds ratio between the two values of the binary variable. A log-odds ratio is simply the natural logarithm of the odds ratio, so $e^{-\beta_i}$ gives us the odds ratio for the ith independent variable. The odds ratio is the proportional increase in the odds of getting a response of 1 when we move the independent variable's value from 0 to 1. Thus, if the odds ratio for an independent variable is r, we are r times more likely to see a response of 1 when the independent variable is 1 than when it is 0. To test for whether a given independent variable has a statistically significant effect on the dependent variable, we can test whether the β's for the independent variable is significantly different from 0. We use the Wald test to do so [137]. The Wald test simply takes a normally distributed test statistic and compares it to a constant value (0, in our case). The Wald test uses the z-distribution (i.e., the standard normal distribution), to determine the probability that the true value of the test statistic is greater than the constant value. We used a general linear model to analyze our time-to-task-completion results. The general linear model encompasses familiar techniques like ANOVA and linear regression, but maintains greater generality. As with logistic regression, the general linear model technique finds coefficients that best fit a function to observed data and then tests whether those coefficients are statistically significant from 0.

We included interface, scale, and an interface*scale interaction as factors in our logistic regression model. The model gave an intercept of 1.52 and coefficients for interface, scale, and interface*scale of -1.22, 0.23, and -0.30. The corresponding odds ratios were 0.29, 1.25, and 0.74. A Wald test of the hypothesis that the interface coefficient was not equal to 0 was significant at the 0.05 level ($Z = 7.43, p < 0.001$), suggesting that there is an effect of interface on accuracy. The direction of the interface coefficient indicates that higher accuracy scores were achieved with the Grid interface. Wald tests of the hypothesis that the other coefficients were not equal to 0 were not significant ($Z = 0.75, p = 0.45$; $Z = -0.80, p = 0.42$), suggesting that the Grid interface's superior accuracy rates compared to the Windows interface are not affected by scale.

We used a general linear model with log-transformed time-to-task-completion as the response and interface, scale, and an interface*scale interaction as factors in the model and found significant effects of interface ($F(1, 450) = 91.15, p < 0.001$), scale ($F(1, 450) = 49.27, p < 0.001$), and interface*scale

($F(1, 450) = 20.81, p < 0.001$) on time-to-task-completion. The directions of the effects indicate that using the Grid (M=53.0 seconds, σ=37.5) led to faster task completion than using Windows (M=88.3 seconds, σ=62.7), that large-scale tasks (M=79.4 seconds, σ=51.8) took longer to complete than small-scale tasks (M=55.2 seconds, σ=49.7), and that the slowdown from small-scale tasks to large-scale tasks was greater for the Grid than for Windows.

6.3.2 Task-by-task results

We followed up the experiment-wide tests with a series of planned tests for significant differences in the accuracy rate and mean time-to-task-completion between the Grid interface and the Windows interface for selected tasks. We conducted these tests based on our hypotheses as to which interface would perform better in each task. Specifically, we hypothesized that we would see more accurate performance using the Grid for the view-simple, view-complex, change-complex, conflict-simple, conflict-complex, and Memogate tasks. We hypothesized that we would see more accurate performance using Windows for the precedence tasks, which were designed to test the drawback in the Grid's new policy semantics. We expected the change-simple and compare-groups tasks to be easy to complete with both interfaces, so we did not test for significant differences in accuracy for those tasks. For the time-to-task-completion data, we hypothesized that the Grid would perform better for all tasks except the precedence tasks, for which we conducted no test because we expected the Grid's drawback to affect accuracy, but to have little effect on time. In all, we had 30 hypotheses to test, but could only test 27 due to insufficient timing data for tasks that were correctly completed by too few Windows participants.

To test our hypotheses, we conducted 14 one-sided Fisher's Exact Tests on the accuracy rates and 13 one-sided t-tests on the mean times-to-task-completion. We log-transformed the time data before testing to ensure that it was normally distributed, as assumed by the t-test. Since we applied 27 total statistical tests, we applied the Benjamini-Hochberg multiple testing correction to keep down the probability of a Type I error in our testing. The correction gave an adjusted per-test α of 0.024, so we considered only tests with p-values at or below this adjusted threshold to be significant.

Results of the tests are summarized in Tables 6.2 and 8.2. The 27 tests yielded 14 significant results, all in favor of the Grid. The 14 significant results included 6 significant differences in accuracy rates and 8 significant differences in mean time-to-task-completion. There was at least one significant result for every task pair except the precedence task pair.

6.4 Discussion

Our results demonstrate that the Expandable Grid interface allows authors to complete tasks more accurately and faster than does the Windows file permissions interface. Because the tasks selected span the range of fundamental policy-authoring operations and cover policies of different sizes, we believe that these results will hold over a broad range of contexts, tasks, and policy sizes in real applications.

Our expectations about the advantages of the Grid's features were borne out by the statistical results of our user study. In particular, our study demonstrates higher accuracy for the Grid compared to Windows for the view-complex, change-complex, and conflict-complex tasks that tested the Grid's *whole policy*, *effective policy*, *group membership information*, *simple changes*, and *new policy semantics* features. The study also

Table 6.2: *Summary of statistical tests for significant differences in accuracy rate for Grids (a_G) and accuracy rate for Windows (a_W) for each task. For all tests except precedence task tests, the hypothesis tested was $a_G > a_W$. For the precedence tests, the hypothesis tested was $a_G < a_W$. The p-values shown are from one-sided Fisher's Exact Tests; p-values below the $\alpha = 0.024$ rejection threshold are shaded and highlighted in bold, indicating significant tests.*

	Small-scale			Large-scale		
Task pair	a_G	a_W	p-value	a_G	a_W	p-value
View-simple	0.89	0.56	$p = 0.03$	0.61	0.56	$p = 0.50$
View-complex	0.94	0.17	$\boldsymbol{p < 0.001}$	1.00	0.39	$\boldsymbol{p < 0.001}$
Change-simple	0.89	0.94	*No test*	1.00	1.00	*No test*
Change-complex	0.61	0.00	$\boldsymbol{p < 0.001}$	0.67	0.17	$\boldsymbol{p = 0.003}$
Compare-groups	0.89	0.83	*No test*	0.67	0.83	*No test*
Conflict-simple	0.67	0.61	$p = 0.5$	0.72	0.61	$p = 0.36$
Conflict-complex	0.89	0.00	$\boldsymbol{p < 0.001}$	1.00	0.06	$\boldsymbol{p < 0.001}$
Memogate	1.00	0.94	$p = 0.5$	0.94	0.78	$p = 0.17$
Precedence	0.89	0.94	$p = 0.5$	0.78	0.78	$p = 0.65$

demonstrates faster times-to-task-completion for tasks that tested these features as well as the small-scale Memogate task, which tested the *visual pop-out* feature.

The precedence tasks were designed to test the drawback (explained above in the "New policy semantics" section) to the Grid's new policy semantics, but we did not see significantly poorer performance with the Grid in these tasks. In these tasks, participants had to allow access to a group but keep an exception that denies access to an individual. Our results suggest that the Grid's policy visualization may mitigate the drawback by helping policy authors realize when they are overriding an exception they wish to keep and allowing them to restore the exception easily.

Our results suggest that the Grid accuracy gains hold up for both small- and large-scale policies. However, our statistical analysis did reveal an interface*scale effect that shows Grid performance slowing down more than Windows performance as policies move from the small to the large scale. We see at least two possible explanations for this effect. First, it may be that Windows participants gained more fluency with their interface than Grid participants did during the small-scale tasks, which, due to limitations in our study design, always came before the large-scale tasks. Second, Windows has in place more advanced search functionality, which may show greater gains as policies get larger. The first explanation may be a point in the Grid's favor, because it suggests that policy authors gained fluency with the Grid quite quickly, so there was little gain to be had through experience. The second explanation suggests that more effort should be put into developing more advanced search functionality for the Grid. Accurately interpreting the interface*scale effect requires further investigation.

While we have shown that Grids outperform the Windows interface for accuracy and speed, it is worth considering some of the limitations of our study method. We gave participants artificial goals in a fictitious scenario with which they were not already familiar. However, in real policy-authoring scenarios, goals often arise intermittently and relate to the author's own environment, with which they are familiar. Thus, some

Table 6.3: *Summary of statistical tests for significant differences in mean time-to-task-completion, in seconds, between successful Grid (G) and Windows (W) participants for each task. For each test, the table shows means (M) and standard deviations (σ) for each interface, and t-statistics (with degrees of freedom (df)) and p-values resulting from one-sided t-tests; p-values at or below the $\alpha = 0.024$ rejection threshold are shaded and highlighted in bold, indicating significant tests.*

	Small-scale			Large-scale		
Task pair	M(σ)	t(df)	p	M(σ)	t(df)	p

View-simple	G:28.8(9.5)
	W:64.6(39.9)
		$-4.37(14)$	**0.001**	G:42.3(9.9)
				W:61.3(25.3)
					$-2.17(14)$	**0.024**

View-complex	G:34.8(19.0)
	W:55.0(28.0)
		$-1.64(2)$	0.12	G:39.1(13.7)
				W:67.4(13.7)
					$-5.11(18)$	**0.001**

Change-simple	G:29.6(29.6)
	W:52.8(24.2)
		$-3.72(25)$	**0.001**	G:50.4(40.3)
				W:42.4(14.4)
					0.13(25)	0.55

Change-complex	G:69.9(19.1)
	W: *Insufficient data*
		–	–	G:100.4(34.6)
				W:143.7(18.3)
					$-3.27(9)$	**0.005**

Compare-groups	G:38.6(14.0)
	W:102.6(36.8)
		$-8.54(28)$	**0.001**	G:111.0(35.8)
				W:126.5(60.0)
					0.43(23)	0.33

Conflict-simple	G:54.7(24.6)
	W:104.1(81.0)
		$-3.23(20)$	**0.002**	G:72.8(41.0)
				W:103.1(64.0)
					$-1.25(18)$	0.11

Conflict-complex	G:29.4(12.2) W: *Insufficient data*	–	–	G:52.4(37.3) W: *Insufficient data*	–	–

Memogate	G:19.8(15.3)
	W:66.1(49.5)
		$-5.41(32)$	**0.001**	G:105.1(42.0)
				W:116.5(72.0)
					$-0.46(28)$	0.33

Precedence	G:42.3(24.6)
	W:117.6(108.2)
		No test	*No test*	G:70.6(29.3)
				W:115.4(59.7)
					No test	*No test*

usability problems that arose in our study, such as searching for files, may not be significant problems in real scenarios, because, for example, authors may already know where all of their own files are stored. On the other hand, real policy authors might encounter usability problems, such as difficulty refining goals, that were not revealed in our study, since we gave participants very specific goals. Another limitation of our study method is that it measures accuracy relative to an artificial goal; a better metric for the efficacy of a policy-authoring interface might be whether it allows authors to write policies that match their own true intentions. In scoring our data to determine accuracy, we ignored any extraneous rules authors made on their way to completing a task (e.g., allowing or denying some access to a user not mentioned in the task statement), because such extraneous rules were rare in our study and usually came about when a participant misinterpreted a task statement. Grid participants created extraneous rules in 8 of 271 tasks that were scored as correct, and Windows participants created extraneous rules in 4 of 183 tasks that were scored as correct, so these extraneous rules would have had little effect on our results if we had scored them as incorrect. In reality, however, such extraneous rules might create serious security vulnerabilities.

In spite of its apparent advantages over the Windows interface, the Grid interface does have room for improvement. We observed three main types of errors participants made with the Grid interface. First were off-by-one errors, in which the participant had lined up the mouse in a desired column, then moved it to find the desired row, only to have the mouse slip, unnoticed, from the desired column into an adjacent column. Participants then made the right changes to the wrong principal. The second common type of error, which was responsible for the lower-than-expected Grid accuracy scores in the conflict-simple tasks, occurred in searching for resources. Some resources in our teaching assistant scenario had similar names, for example, a folder called “Assignment 1” and a file called “assignment1.pdf.” Some participants, when searching for the keyword “assignment,” came across the “assignment1.pdf” file first and assumed they had found the correct resource, although the task required changes to the folder. The third common type of error resulted from the 90-degree angle of the text in the tree at the top of the Grid interface. Participants reported that this rotated text was difficult to read. This error occurred, for example, in our small-scale compare-groups task, in which participants had to see if there was any overlap between 2007’s TAs and 2006’s students. Some participants correctly read who the TAs were, and correctly opened the students group to view its members, but simply overlooked the name of the overlapping member, even though it was visible.

These three errors suggest improvements that could be made to the Grid interface. Several participants explicitly asked for the ability to lock the Grid interface’s crosshairs on a given row or column, so that they could then find a desired item in the other dimension without the mouse slipping off the locked row or column. Also helpful in this regard might be some variant on a focus + context visualization, in which a desired region of the grid could be made much larger than its surroundings. A focus region would make for a larger mouse target, and hence for more accurate mouse positioning. The second error could be addressed with improved search features. Although our Grid interface has a search bar and supports search, its display of multiple search hits is inconspicuous, and was overlooked by many users. If they searched for the keyword “assignment,” there would be two hits, but they may have noticed only one. In contrast, Windows has a much better developed search feature for displaying multiple search hits—it shows all search hits in a list with summary details of each hit. Only one Windows participant made the error of mistaking the “assignment1.pdf” file for the “Assignment 1” folder compared with five Grid participants who made this error. Finally, the vertical text in the Grid interface is too difficult to read. A number of solutions may address this problem: changing the angle to 45 degrees would probably help somewhat, the letters could be rotated so that words could be read top-to-bottom, or there could be an option to rotate the whole tree to be horizontal when it is not needed to label the grid columns.

6.5 Conclusion

The results of our user study demonstrate that our Expandable Grid interface for authoring file permissions policies is superior to the Windows XP native file-permissions interface. The results also strongly suggest that the Grid's superior performance was due to its features that were designed to overcome deficiencies in the list-of-rules model. We have thus provided strong evidence that, from a usability perspective, the Expandable Grid approach to policy-authoring interface design is preferable to the list-of-rules approach.

CHAPTER 7

Policy Semantics for the File Permissions Expandable Grid Interface

The policy semantics that underlies a policy-authoring user interface may be as important to usability as the interface itself [23, 138]. A policy semantics is the method by which rules are converted into policy. In our design of the file permissions Expandable Grid interface, we changed the way rule conflicts are resolved in policies authored with our interface (see Section 6.1.1). This change to the semantics, in which conflicts are resolved by choosing the most recently created rule, allowed authors in the Jana task to simply click on the red boxes corresponding to Jana's READ and WRITE permissions to turn them green. The Grid interface, combined with the semantics change, led to 100% accuracy for the Jana task in our user study amongst participants using the Grid interface compared to 6% amongst those using the Windows interface. This result raises the question, however, of whether the Grid's improved accuracy rates were the result of the Grid's presentation aspects (namely, the interactive matrix display of the full, effective policy), the revised semantics, or both. The first of two studies described in this chapter finds that the answer is, in fact, both. The study was an extension of the study described in Chapter 6. In the extension study, we observed 18 participants performing the same tasks in a new condition: an Expandable Grid interface with the Windows semantics. We found that for tasks that do not involve rule conflicts, for which the semantics makes no difference, performance with the Grid with Windows semantics was still better than performance with the Windows interface. For tasks in which the semantics does make a difference, the Grid with Windows semantics performed better than the Windows interface but not as well as the Grid with our revised semantics.

The extension study result raises an interesting question: if usability can be improved by changing the semantics, what semantics is best from a usability perspective? In this chapter, we shed some light on this question. We describe a specificity semantics, in which rules applying to more specific entities take precedence over rules applying to less specific entities. The specificity semantics includes some of the dynamic aspects that were missing from the revised semantics we used in the prior study, namely, how the policy changes when users are added to or removed from groups or files are added to or removed from folders. We then describe a second user study that compared three interface conditions: the Expandable Grid with the specificity semantics, the Expandable Grid with the Windows semantics, and the Windows interface (with the Windows semantics, of course). The user study involved new tasks, not included in the prior study, that exercise some of the dynamic aspects of the specificity semantics. Results of the study indicate that which semantics is better depends on the specific task an author is trying to perform, but that, no matter which semantics is in use, the Grid presentation leads to better performance than the Windows presentation.

7.1 Problem description

Our objective is to find a semantics that makes sense to policy authors. By "a semantics that makes sense," we mean a semantics for which the output decisions most closely match those intended by the author. The Jana task, as described in Section 1.3.1, is an example of a situation in which the Windows DENY decisions do *not* match those intended by the author. In the Jana task, Jana is a member of two groups, one of which is allowed and the other of which is denied access to a file she is supposed to be able to access. When participants in our study created an explicit rule stating that Jana is allowed access, they expected she then actually had access. In fact, because of the deny-precedence semantics, the group rule denying Jana access was still in effect.

It is not possible to create a consistent semantics that always matches authors' intentions, since authors may want different outputs in situations that, from a syntactic point of view, are indistinguishable. For example, an author may want rule conflicts for one user to be resolved by always issuing DENY decisions, but rule conflicts for another user to be resolved by always issuing ALLOW decisions. However, we believe authors may have systematic biases in their intentions and that it is possible to adapt a semantics to such biases.

It may be the case that different semantics are best suited to different policy-authoring user interfaces. Our objective is to find a semantics that is well suited to the file permissions Expandable Grid interface.

7.1.1 Access-control model

Policy semantics can only be considered within the context of an access-control model. In this section, we introduce our file-system access-control model, which is a simplified version of the Windows NTFS file system and is similar to other file and access-control systems in real use.

Our access-control model is built on top of a file-system model. The file-system model has *resources*, which include both files and folders that contain files, and *principals*, which include both users and groups that contain users. Principals may access resources in different ways, which we call *actions*. Actions in our file-system model include READ, WRITE, EXECUTE, DELETE, and ADMINISTRATE. The *access-control system* is that part of the file-system model that determines which principals may perform what actions on which resources. An access-control system receives *requests* for access to resources and issues *decisions* in response. Requests are 3-tuples consisting of a user, file, and action (e.g., (jsmith, "file.txt," READ)). A folder serves both as a container of files and as an accessible resource in its own right. In our model, we consider each folder to contain a file called '.', which represents the folder in its role as an accessible resource. This special file could be used as the resource in a request, but folders as containers cannot. Hereafter, when we refer to a folder, we mean a folder in the container sense. Decisions are literals from the set {ALLOW, DENY} and indicate whether access will be granted or not. The access-control system maintains a *rule base*, which is a set of *access rules*, *group rules*, and *folder rules*. The access-control system reaches access decisions by an algorithm that takes a request and a rule base as input, consults the rule base, and issues a decision as output.

Access rules (or simply *rules*) state that some 3-tuple consisting of a principal, resource, and action should be mapped to either ALLOW or DENY. If an access rule's principal is p, we say the access rule *covers* p; similarly, if the access rule's resource is r, we say the access rule *covers* r. Note that valid 3-tuples in access rules are a superset of valid requests: a valid request cannot contain a group or a folder, but the 3-tuple in a rule can. Access rules whose decision is ALLOW are called ALLOW *rules* and access rules whose decision is DENY are called DENY *rules*. *Group rules* state that a user is a member of a group. *Folder rules* state that a file (or subfolder) is contained within a folder. Note that, in our model (as well as in Windows), groups cannot contain other groups, but a single user can appear in any number of groups, so group memberships may overlap. The file structure, however, is strictly hierarchical, so a file may only appear in one folder; each file and each folder has exactly one parent, except the root, which has no parent.

If the rule base simply consisted of exactly one access rule for each request, mapping a request to a decision would be a simple matter of looking up the rule for the request and outputting the specified decision. In reality, however, specifying an access rule for every possible request would be a huge burden on an access-control policy author, so the rule base will have access rules whose principals are groups (we call such access rules *group access rules*) and access rules whose resources are folders (we call such access rules *folder access rules*). Group access rules apply to all users in the specified group and folder access rules apply to all subfolders and files in the specified folder. Note that a rule can be both a group access rule and a folder access rule.

When the access-control system receives a request, it consults the rule base to find all access rules that *apply to* the request. An access rule applies to a request if:

- The access rule's principal is the same as the request's user, or is a group containing the request's user; and
- The access rule's resource is the same as the request's file, or is a folder containing the request's file; and
- The access rule's action is the same as the request's action.

It may rarely be the case that exactly one rule applies to a request. Sometimes, no rule will apply. The access-control system has a *default rule* that determines the decision that will be issued when no access rule applies. In our model, the default rule is a default DENY, so a DENY decision will be issued when no access rule applies.

Sometimes more than one access rule may apply to a request, in which case access rules can be in conflict with each other. If two access rules apply to a request and one access rule's decision is ALLOW and the other's decision is DENY, the rules are in conflict. A conflict may arise, for example, when a user is a member of two groups, say, GroupA and GroupB. If one access rule states that GroupA is allowed read access to a file and another access rule states that GroupB is denied read access to the file, there is a conflict.

By *semantics*, we mean the algorithm that maps requests to decisions to form the effective policy. We call the mapping of all possible requests to decisions the *effective policy*. A single mapping of a request to a decision is an *effective permission*. In this chapter, we will discuss multiple semantics, but all are variants on the following algorithm that takes a request req as input and returns a decision:

1. Create a set $ruleset$ and add to it all rules in the rule base that apply to req;
2. If $ruleset$ is empty, return DENY;
3. If $ruleset$ contains exactly one rule, return that rule's decision;
4. If all rules in $ruleset$ have the same decision, return the rules' common decision;
5. If there are conflicting rules in $ruleset$:
    (a) For each ALLOW rule in $ruleset$:
        i. For each DENY rule in $ruleset$:
            A. Match the ALLOW rule against the DENY rule according to a *conflict-resolution method*, which will vary by semantics;
            B. If the ALLOW rule loses the match, break and proceed to the next ALLOW rule;
            C. If the ALLOW rule wins the match, continue with the next DENY rule;
        ii. If an ALLOW rule wins its match against every DENY rule, return ALLOW;
    (b) If no ALLOW rule wins against every DENY rule, return DENY.

When a rule wins the match in step 5(a)iA, we say it *takes precedence* over the rule it was matched against. When the algorithm returns a rule's decision for a request, we say the rule is *in effect* for that request.

The interesting part of the algorithm outlined above is the conflict-resolution method that chooses a winner between two rules in conflict. The conflict-resolution method is the main aspect in which the semantics we propose in this chapter differs from the Windows semantics.

7.1.2 Desired properties in a semantics

We have identified the following desirable properties in a policy semantics for a policy that is to be authored with an Expandable Grid. The first of these properties, direct manipulation, is based on our user studies that have demonstrated usability problems when the property does not hold (for example, notice results for the Jana task in Section 6.3); the second and third of the properties, exception-rule preservation and order independence, are based on our intuition about how to minimize policy-authoring errors and their impact; and the fourth, fail safety, is a widely accepted computer security design principle.

- *Direct manipulation*: Direct manipulation is a user-interface design principle stating that interfaces should allow users to operate directly on objects or data of interest [121]. In policy-authoring interfaces, direct manipulation translates to allowing authors to change effective policy directly whenever possible. For a Grid-based interface to support direct manipulation, the author should be able to go to the cell corresponding to the principal, resource, and action for which they wish to set policy, and set the policy right in that cell, without having to go elsewhere. Put another way, when the author clicks on a green cell, it should turn red and when the author clicks on a red cell it should turn green.
- *Exception-rule preservation*: We define an *exception rule* of an access rule as another access rule that covers a subset of the principals and resources covered by the first and that has a different decision

from the first. For exception-rule preservation to hold, exception rules should stay in effect even when rules covering a superset of the exception rules' resources and principals are specified. We designed the Imani task, described in Section 6.1.1, to illustrate the problem that can arise with exception rules. In the Imani task, Imani is a TA who was suspected of selling answers on an exam and was denied access to the Answers folder. The Imani task then asks the participant to grant all of the TAs access to the Answers folder, but, of course, Imani should still not be given access. In other words, the rule denying Imani access to the Answers folder is an exception rule to a new rule that should grant all the TAs access. The Imani task calls for the exception rule to stay in effect. In semantics that violate exception-rule preservation, new rules might take precedence over existing rules. If the Imani task were attempted with such a semantics, a new rule allowing all of the TAs access to the Answers folder might override the existing exception rule, in which case Imani would be allowed access unless the author remembered to explicitly re-assert the exception rule. In our study, most participants had no difficulty completing the Imani task, since we explicitly told participants to preserve the exception rule. However, to see how serious this situation could be in a real scenario, imagine an author who set an important exception rule weeks earlier but has forgotten about it, and now sets a new rule that overrides it. The author may not remember the existence of the exception rule and would likely fail to notice it is now being overridden.

- *Order independence*: The order in which operations (such as setting access rules or changing group or folder memberships) are performed should not affect the effective policy. It may be difficult for authors to remember to perform operations in a particular order, and the Grid does not indicate the order in which operations have been or should be performed, so if effective policy depends on order, it may be difficult for authors to know how to achieve the effective policy they want. For example, consider trying to complete the Imani task with a semantics in which rule conflicts are resolved by letting the most recently set rule take precedence over the other rules in conflict. Now if an author sets a rule stating that TAs have access to the Answers folder, then sets the exception rule stating that Imani is denied access, the author correctly completes the task, because the more recent rule applying to Imani will take precedence over the original rule applying to the group. If, however, the author first sets the exception rule stating that Imani is denied access, then sets the rule stating that the TAs are allowed access, the author incorrectly completes the task, because the more recent rule applying to the group will take precedence over the original rule applying to Imani. If an author is unaware that order matters, and sets rules in an arbitrary order, errors are very likely. Moreover, even an author aware that order matters may have forgotten the order in which rules were set previously. In this case, the author may not know why the effective policy turned out the way it has or how to fix it.

- *Fail safety*: In general, false rejects (cases where authorized access is denied) should be preferred to false accepts (cases where unauthorized access is allowed). False rejects can be corrected by changing the policy to allow the desired access, but once a false accept has caused data to be released to an unauthorized party, the data cannot be taken back. Saltzer and Schroeder, in a seminal work on computer security, list fail safety as one of their eight design principles for protection mechanisms [113]. For a semantics, preserving fail safety means favoring DENY rules over ALLOW rules in the absence of any better reason to favor ALLOW rules.

Note that no semantics can satisfy all properties in all situations, since some of the properties would lead to contradictory decisions in the same situation. For example, fail safety will always call for conflicts to be resolved by issuing DENY decisions, but direct manipulation and exception-rule preservation may call for some conflicts to be resolved by issuing ALLOW decisions.

7.1.3 Conflict-resolution methods

To resolve rule conflicts, there must be a method for unambiguously choosing a decision. Most conflict-resolution methods in practice choose one of the rules in conflict to take precedence over the others. (Other methods are possible, however, such as "majority rules"—choosing the decision of the majority of the rules in conflict. It is also possible to disallow conflicts through the use of static checking, in which any changes to the rule base that introduce rule conflicts are recognized and disallowed at the time the rule is added. While static checking is a possible approach to improving policy-authoring usability, we leave consideration of static checking to future work.) Some of the possible conflict-resolution methods for choosing rules to take precedence are:

- *Specificity precedence*: A rule that covers a more specific entity takes precedence over a rule that covers a more general entry. For example, a rule that covers a user takes precedence over a rule that covers a group, or a rule that covers a subfolder or file takes precedence over a rule that covers the subfolder's or file's parent.

- *Deny precedence*: Rules whose decision is DENY take precedence over rules whose decision is ALLOW.

- *Order precedence*: Rules are totally ordered, usually by the policy author, so the author can explicitly state which rules take precedence over others. This method is commonly used in firewall policies.

- *Recency precedence*: Rules specified more recently in time take precedence over rules specified less recently in time. Note that recency precedence is equivalent to order precedence where order is determined by the time at which each rule was set.

These conflict-resolution methods may be used in combination. It is possible to use different conflict-resolution methods depending on whether conflicting rules differ in the principals they cover, the resources they cover, or both. For example, the Windows NTFS semantics uses deny precedence if conflicting rules differ in principals, but specificity precedence if conflicting rules differ in resources or in both resources and principals. It may also be necessary to resort to multiple conflict-resolution methods when one method fails to resolve a conflict. For example, when conflicting rules cover groups, but those groups are peers of each other, specificity precedence cannot resolve the conflict.

It is worth noting that conflicts are not inevitable in a rule-based access-control system. It is possible for an access-control system to avoid rule conflicts by only allowing policy authors to specify exceptions to the default rule, and not allowing exceptions to the exceptions. For example, if the default rule is DENY, and policy authors can only specify requests that should be allowed, no conflicts will arise. However, such a system can be unwieldy [120]. For example, if all members of a large group of principals except one are to be allowed access to a file, it would be necessary to form a new "all-but-the-exception" group or to specify a rule allowing access for every single member of the large group except for the one. Access-control systems where situations such as these may arise typically allow policy authors to specify both ALLOW and DENY rules; hence, rule conflicts may arise in which author-specified rules conflict with each other.

7.1.3.1 Pros and cons of different conflict-resolution methods

Each of the listed conflict-resolution methods has pros and cons. Pros take the form of desired semantics properties from section 7.1.2 that the method supports, and cons take the form of desired semantics properties that the method does not support. We discuss the pros and cons of each method below.

Specificity precedence

Specificity precedence supports exception-rule preservation and order independence, but it can lead to violations of direct manipulation and fail safety. Exception rules are more specific versions of the rules that cover a superset of the exception rules' principals and resources, so specificity precedence naturally ensures that exception rules will stay in effect. Specificity precedence does not depend on the order in which rules were set, so it maintains order independence. When a DENY rule and an ALLOW rule conflict, specificity precedence makes it possible for the ALLOW rule to take precedence, so if the ALLOW rule was not intended to apply, a false accept could occur, violating fail safety. Specificity precedence maintains direct manipulation for whenever a rule is being specified at the user or file level, but direct manipulation may be violated when a rule is being set at the group or folder level. For example, if a policy author intends to allow all users in a group to read a file f, but the group contains one user u covered by a DENY rule for f, then the author's attempt to turn the group's Grid square for f green will fail—the square will remain yellow, indicating that the effective policy has both ALLOW (for most users in the group) and DENY (for u) decisions for users in the group.

Deny precedence

Deny precedence emphasizes fail safety over all the other properties, although it also maintains order independence. When DENY rules always take precedence over ALLOW rules, unintended false accepts as the result of rule conflicts cannot occur, so fail safety is maintained. However, direct manipulation in the Grid can be violated. For example, suppose a DENY rule covers a group g and a file f, but the author wishes for a user u within g to be allowed to access f. Now, if the author clicks on the Grid square for u and f, intending to turn the square green, the square will remain red, since the group DENY rule will take precedence. Exception-rule preservation may be violated in a similar, but chronologically reversed, situation. For example, suppose there is an ALLOW rule that covers a user u and a file f, and the author wishes to set a DENY rule that applies to a group g containing u, but to allow the ALLOW rule at the user level to remain in effect. The ALLOW rule cannot remain in effect under deny precedence.

Order precedence

Order precedence places the burden of resolving rule conflicts in the hands of the policy author, so it is possible for any of the four properties to be violated. The Grid does not currently have a mechanism to support direct ordering of rules by the author, so we do not further consider order precedence here, since our objective is to find a semantics that is well suited to the Grid. Order precedence is perhaps best suited to list-of-rules interfaces, since a list naturally shows rules in order. If order precedence must be used for a particular policy domain (perhaps because it gives authors a high degree of flexibility to resolve conflicts), a list-of-rules interface might be best. Even so, list-of-rules interfaces for policies using order precedence might be enhanced by a Grid-based display of the effective policy.

Recency precedence

Recency precedence supports direct manipulation but can lead to violations of exception-rule preservation, fail safety, and order independence. By definition, under recency precedence, when the author specifies a rule, it immediately takes precedence over existing rules, so direct manipulation is naturally supported while exception-rule preservation is naturally violated. Recency precedence can result in a more recent ALLOW rule taking precedence over an intended DENY rule, so can lead to false accepts, violating fail safety. Recency precedence also orders rules based on the chronological order in which they were specified, so it violates order independence. Section 7.1.2 explained how recency precedence can violate fail safety and order precedence.

7.1.4 Weaknesses of Windows NTFS semantics

The Windows NTFS access-control semantics combines specificity and deny precedence. When conflicting rules differ only in the resources they cover, specificity precedence is used. When conflicting rules differ only in the principals they cover, deny precedence is used. When conflicting rules differ in both the principals and the resources they cover, specificity precedence is used.

The Windows semantics has three primary weaknesses we seek to improve upon. First, it violates direct manipulation and makes it very difficult to resolve certain rule conflicts; second, it behaves in a non-intuitive way in the presence of what we call two-dimensional conflicts, where rules in conflict differ in both the principal and resource dimensions; finally, it behaves in a potentially unexpected way following a file move.

Since the Windows semantics has the weaknesses of both specificity and deny precedence, it can lead to violations of direct manipulation, exception-rule preservation, and fail safety. We focus on improving the Windows violations of direct manipulation in the semantics we propose below, because these violations are the most difficult part of the Windows semantics for authors to deal with. The Jana task is a good example of where the Windows semantics leads to violations of direct manipulation. Jana is a member of two groups, one of which is allowed access to a file and the other of which is denied access to that file. The Windows semantics uses deny precedence to resolve the conflict, so Jana is effectively denied access to the file. There are two ways to grant Jana access: the DENY rule applying to the latter group can be removed, or Jana can be removed from the group. Direct manipulation is violated because the task cannot be completed by simply manipulating rules applying to Jana; the task requires making a change at the group level. Moreover, both potential solutions may be unsatisfactory. Removing the DENY rule may change policy for the other members of the group and for members added to the group in the future. Removing Jana from the group may be undesirable since it may affect Jana's permissions on other resources. The Windows semantics provides no entirely satisfactory way to complete the Jana task.

Besides violations of direct manipulation in the presence of rule conflicts, we seek to improve upon another aspect of the Windows semantics conflict resolution—behavior in the presence of a two-dimensional conflict. A *two-dimensional conflict* occurs when two rules are in conflict and one rule is more specific in the principal dimension while the other is more specific in the resource dimension. For example, in the Lance task we describe in Section 7.5.1.3, one rule denies Lance access to an Admin folder, but another rule allows a group he is in access to the gradebook.xls file contained in the Admin folder. Since neither rule is strictly more specific than the other, specificity precedence cannot resolve this conflict. Windows resolves such conflicts by favoring the resource dimension over the principal dimension, so the rule that is more specific in the resource dimension will take precedence. Favoring the resource dimension seems arbitrary, however, and may confuse policy authors.

Finally, we seek to improve upon the Windows semantics' behavior following a file or folder move. In NTFS file systems, when a file or subfolder is moved from one parent folder to a new parent folder on the same volume, the moved resource continues to inherit rules from the old parent folder instead of immediately inheriting the rules of the new parent [90]. Only when a change is made to the access-control list of the moved resource or the new parent does the moved resource inherit rules from the new parent folder. This resource-move semantics could lead to confusion when an author assumes that a moved resource will inherit its new parent's permissions, which seems an intuitive assumption to make. Even more confusing is that making a change to one principal's access rules for a folder could suddenly and unexpectedly change the access for another principal to a file or subfolder within the folder. For an example of what can go wrong with the Windows semantics in the presence of a resource move, see the description of the Assignment task in Section 7.5.1.3.

7.2 Proposed solution

We have already tried to address some of the weaknesses of the Windows semantics with the semantics we described in Section 6.1.1. However, that semantics, which uses recency precedence, also has drawbacks. It violates order independence and, worse, its behavior in the presence of dynamic changes to principal and resource structure (such as file moves and group membership changes) was unspecified. In this section, we describe a semantics based on specificity precedence that we believe retains the primary advantage of recency precedence (namely, its direct-manipulation solution to the Windows rule-conflict problem) while maintaining exception-rule preservation and order independence. We also specify the behavior of our new semantics in the presence of dynamic principal and resource structure changes.

7.2.1 Our proposed specificity semantics

We propose a semantics that we believe will address the weaknesses of the Windows semantics and will make sense to policy authors using the Grid. We intend for the semantics to make sense not only when the principal structure and the file hierarchy are static, but also in the presence of group membership changes, user and file creation and deletion, and file moves.

7.2.1.1 Conflict-resolution method

We propose to use specificity precedence to resolve conflicts when possible and to resort to deny precedence only when specificity precedence fails. By using specificity precedence, we address the rule-conflict weakness in Windows in which the semantics violates direct manipulation. Specificity ensures support for direct manipulation for rules that cover users and files. Thus, tasks like the Jana task are easily completed using the Grid with specificity semantics. The rule-conflict weakness in Windows where either a group access rule has to be changed or Jana has to be removed from the group is solved by allowing a specific rule that applies to Jana to take effect. Specificity precedence also supports exception-rule preservation and order independence, so it has advantages over recency precedence, which does not support these properties. Specificity's support for exception-rule preservation comes at the cost of direct manipulation at the group and folder level, but we believe these problems with direct manipulation can be mitigated with the interface solutions described in Section 7.2.2.2. Specificity precedence's advantages in exception-rule preservation and user- and folder-level direct manipulation come at the cost of potential violations of fail safety (since a more specific ALLOW rule can take precedence over a less specific DENY rule), but this tradeoff is unavoidable. Our proposed semantics does its best to maintain fail safety by resorting to deny precedence when specificity precedence is insufficient to resolve conflicts, as when peer group-level rules are in conflict.

To address the weakness of the Windows semantics in the case of a two-dimensional conflict, our proposed semantics uses deny precedence for two-dimensional conflicts. Since there is no natural way to resolve such conflicts using specificity, we ensure fail safety by using deny precedence. We believe the Windows semantics' use of specificity in the resource dimension is arbitrary and likely to be confusing to policy authors. Our choice of deny precedence may or may not be confusing, but at least it is safe and, in any case, it is easily overridden by creating a rule that is more specific in both resources and principals than the rules in conflict.

7.2.1.2 Dynamic changes

The behaviors of our semantics for all possible dynamic principal and resource changes are as follows:

- *Create group or user*: New groups and users start initially with no access rules, so all decisions default to DENY.
- *Delete user*: A deleted user is denied access to everything, and its access rules are removed from the rule base.
- *Delete group*: When a group is deleted, all of its access rules are removed from the rule base, so they no longer apply to users who were in the group.
- *Add user to group*: When a user is added to a group, the user inherits all of the group's access rules.
- *Remove user from group*: When a user is removed from a group, the group's access rules no longer apply to the user.
- *Create file or folder*: New files and folders inherit rules applying to their parent in the file hierarchy.
- *Delete file*: When a file is deleted, its access rules are removed from the rule base and the file can no longer be accessed.
- *Delete folder*: When a folder is deleted, its contents are also deleted, so no one can access the folder or the files it contained.
- *Move file or subfolder to new parent folder*: When a file or subfolder is moved to a new parent folder, it immediately inherits the new parent folder's access rules.

Note the behavior in response to a moved file or folder, which is different from the Windows semantics' behavior. A moved file or subfolder immediately inherits rules applying to its new parent and gives up rules inherited from its old parent. This behavior addresses what we believe is a confusing and potentially error-inducing aspect of the Windows semantics.

Our semantics is fairly straightforward in the presence of dynamic principal and resource changes. This straightforwardness is in contrast to (and, we believe, an improvement upon) a recency precedence semantics, in which it is unclear in what access rules will take precedence. To give an example of an ambiguous case that arises when using a recency semantics: If a user is added to a group, should that group's access rules immediately take precedence over all of the user's existing access rules? Or should the group's access rules only take precedence if they were set more recently than the user's existing rules? In our semantics, the chronological ordering of rules does not matter, so ambiguities as to what where to place a rule in order do not arise.

7.2.2 New Grid interface features

Our new semantics requires some new features to be added to the file-permissions Expandable Grid interface. First, we need functionality for making dynamic changes, i.e., creating and deleting users, groups, files, and folders, changing group memberships and moving files and subfolders. Second, since we allow these dynamic changes, it is necessary to show which principals and resources rules have been set for.

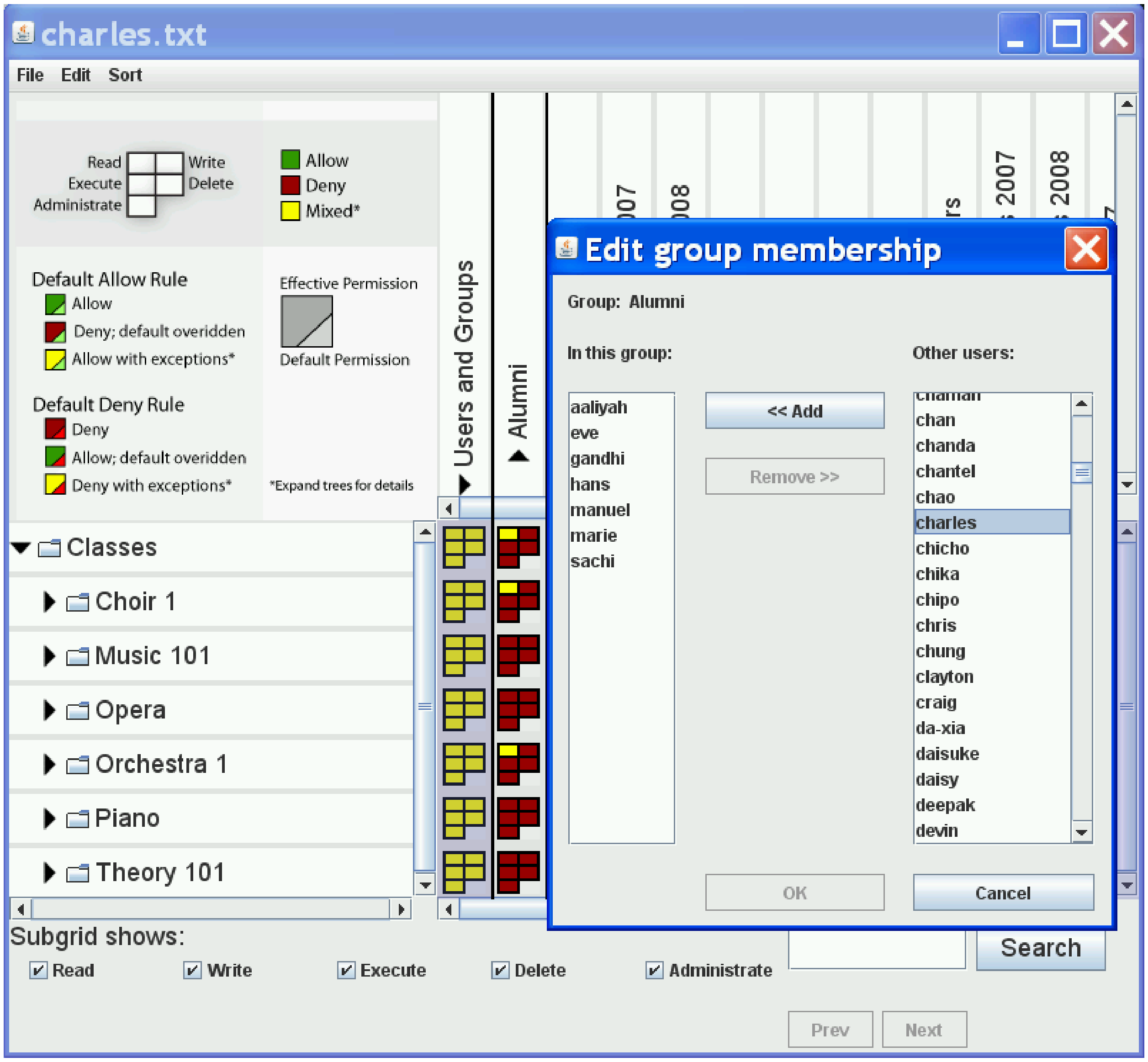
7.2.2.1 Dynamic changes

We added functionality to our file-permissions Expandable Grid interface to allow policy authors to create and delete users and groups, edit group memberships, and move files between folders. Dialogs to create and delete users and groups can be accessed from the menu bar and the dialog to edit group memberships (see Figure 7.1) can be accessed by right-clicking on a principal. Files can be dragged and dropped to move them from one folder to another.

7.2.2.2 Showing rules explicitly

In the file-permissions Grid developed for the study in Chapter 6, the Grid showed effective policy only—rules existed in the rule base, but were not shown to authors. However, in the presence of dynamic changes, authors may need to know what will happen to the effective policy when they add a user to a group or create a file within a folder, for example. Moreover, showing the rules explicitly may help mitigate the primary drawback of our proposed semantics, which is that direct manipulation is violated at the group- and folder-level. For example, if the author is trying to set a new rule that covers a group, but there are existing exception rules to the new rule, the new rule will not take precedence over the existing rules. The effective policy will issue the group access rule decision for some group members but the exception-rule decisions for other group members. Thus, there will be a difference in the decision for the group access rule and the the effective policy for the group—the decision will be ALLOW or DENY, but the effective policy may be mixed ALLOW and DENY. The Grid, if it were to show only one square of color per group, resource, and action, would have to choose between showing the group access rule—which authors may need to see in order to know that they have specified a group access rule they wanted—or showing effective policy—which the Grid was designed to show. We get around this problem by showing two colors in subgrid squares where an access rule has been set.

In subgrid squares corresponding to a principal, resource, and action for which an access rule has been set, we draw a small triangular tab in the lower left-hand corner. We refer to this tab as a "dog-ear." The dog-ear is light green to represent an ALLOW rule or light red to represent a DENY rule set for the specific tuple of principal, resource, and action. We color the remainder of the subgrid square according to the effective permission, as usual. We added a new legend to the Grid to show all possible combinations of effective permission color (green, red, or yellow) and dog-ear color (light green, light red, or none). Note that a square can have a light green dog-ear on a red background if there is an ALLOW rule for the group,



Group: Alumni

In this group: aaliyah, eve, gandhi, hans, manuel, marie, sachi

Other users: chan, chanda, chantel, chao, charles, chicho, chika, chipo, chris, chung, clayton, craig, da-xia, daisuke, daisy, deepak, devin

Figure 7.1: *A screenshot of the Grid interface with the edit group membership dialog showing. The dialog allows the policy author to add users to a group or remove them from the group.*

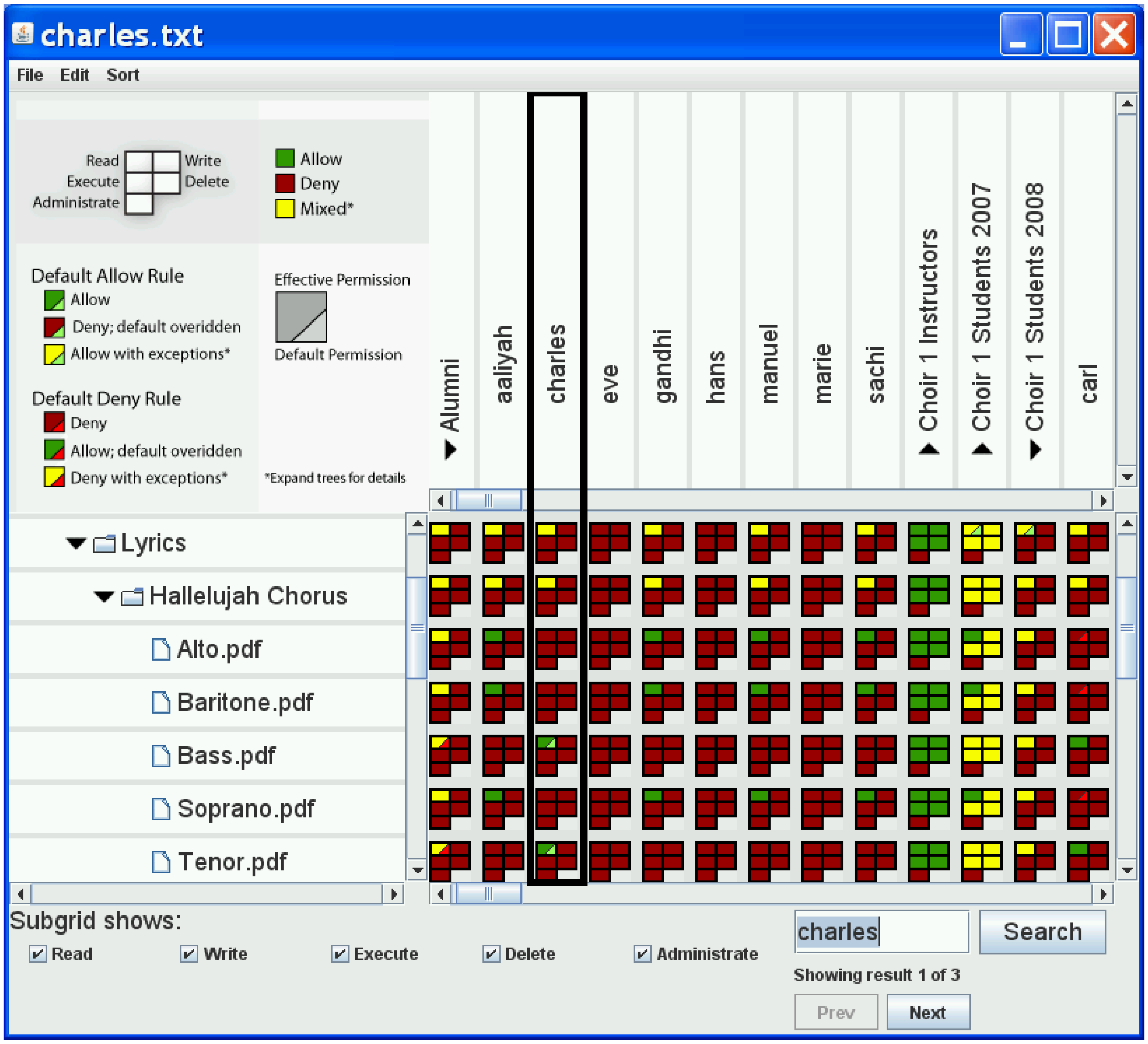


Figure 7.2: *A screenshot of the Grid interface for the Charles task, after Charles has been moved to the Alumni group. Dog-ears are visible in 9 squares in this screenshot, such as the squares corresponding to Charles's* Read *access to the Bass.pdf and Tenor.pdf files.*

but all members of the group have been explicitly denied access. Similarly, a square can have a light red dog-ear on a green background. Dog-ears and the new legend can be seen in Figure 7.2.

The new legend presented at least one design challenge of note, namely deciding on terminology for "access rule." The legend shown in Figure 7.2 uses the terms "default allow rule," "default deny rule," "default," and "default permission" to refer to what we have called "ALLOW rule," "DENY rule," and "access rule" (the term "default" is simply a shortened form of "default permission," and both refer to access rules) in this document. We did not use the same terms in the legend that we have used in this document because we expected our targeted demographic of novice and occasional policy authors to be confused by the distinction between access rules and effective permissions. We thought, for example, that the use of the term "ALLOW rule" for requests where access may not actually be allowed would be very confusing. We hoped that the idea of a "default" rule would help clarify the confusion by suggesting a rule that will take effect unless another rule overrides it. (Note, therefore, that the use of the term "default rule" in the legend is not the same as its use in this document, where it refers to the decision that will be issued when no access rule applies to a situation.) Nevertheless, in a user study to be described below, many participants using the Grid with the new legend remained confused about the distinction between access rules and effective permissions, so our use of "default" in the legend seems not to have been effective. We have yet to determine a better alternative; we propose "set rule," "explicit rule," and "stated rule" as possibilities to try in future work.

7.2.3 Summary of differences between our semantics and Windows

Our specificity semantics is different from the Windows semantics in three primary ways. First, our semantics allows a rule with a more specific principal to take precedence over a rule with a less specific principal. Second, our semantics uses deny precedence in two-dimensional conflicts. Finally, our semantics propagates rules from a new parent to a moved resource immediately upon the move. The first two of these differences can be seen in Table 7.1, which lays out the entire space of rule conflicts with respect to the structural relations between principals and resources of the rules in conflict. In a conflict, there will be an ALLOW rule in conflict with a DENY rule. The differences between our semantics and the Windows semantics can be seen in two of the table cells: first, where the ALLOW rule's principal is contained by (and thus is more specific than) the DENY rule's principal, but the two rule's resources are the same; second, where the ALLOW rule's principal contains the DENY rule's principal, but the ALLOW rule's resource is contained by the DENY rule's resource.

7.3 Related work

There is an extensive body of work describing formal access-control models and languages for various contexts. The authors of these models and languages necessarily describe a semantics and usually address conflict resolution. Most of these works are concerned with describing the formal aspects of their models and proving theoretical properties about them. A few of these works address ease of policy-authoring under their semantics, but none that we are aware of actually ran user studies to evaluate their semantics empirically as we have. We survey some of these works on formal access-control models below, with an emphasis on those that address usability issues and conflict-resolution methods.

Shen and Dewan present an access control model for collaborative environments [120]. They list, as one

Table 7.1: *Table showing which rule, of an* ALLOW *rule and a* DENY *rule in conflict, will take precedence for our specificity semantics (S) and Windows semantics (W). The table shows the relevant cases defined by the rules' principals and resources. Each cell shows which rule takes precedence and the conflict-resolution method in play: specificity in the* resources*, specificity in the* principals*, specificity in* both *resources and principals, or* deny *precedence. In the case where both rules' resources and principals are the same, there will be* no conflict*, since the more recently set rule will have overwritten the other.*

		Relationship between ALLOW rule's principal and DENY rule's principal			
		Contains	Peer	Same	Contained by
Relationship between ALLOW rule's resource and DENY rule's resource	Contains	DENY (both)	DENY (resources)	DENY (resources)	S:DENY (deny) W:DENY (resources)
	Same	S:DENY (principals) W:DENY (deny)	DENY (deny)	no conflict	S:ALLOW (principals) W:DENY (deny)
	Contained by	S:DENY (deny) W:ALLOW (resources)	ALLOW (resources)	ALLOW (resources)	ALLOW (both)

of their requirements for their access-control model, "easy specification" of access-control policies. Their model includes both ALLOW and DENY rules (which they call "positive rights" and "negative rights"), thus it can have conflicts. They provide an argument that including DENY rules in their model improves usability; their argument is similar to our argument that some policy-specification tasks become quite unwieldy without DENY rules. For conflict resolution, they use specificity precedence when possible and order precedence otherwise. Dewan and Shen present a similar access control model for multiuser interfaces [42]. Dewan and Shen provide for specification of an inference function that controls how conflicts are resolved and allows for much flexibility in specifying a conflict-resolution method. Deny precedence, specificity precedence, and allow precedence (in which ALLOW rules take precedence over DENY rules) are all possible under this model. They acknowledge the two-dimensional conflict issue. They suggest resolving such conflicts by choosing a prime dimension and a subordinate dimension, and allowing the prime dimension's conflict-resolution method to be applied first.

Kapadia et al.'s Know system addresses usability for policy-based systems [67]. Know sits within an access control system and provides feedback when access is denied. The feedback can improve policy-authoring usability by explaining why access was denied and how access might be obtained. Kapadia et al. provide for meta-policies that control what portions of a policy can be revealed in providing feedback.

Sampemane et al. describe an access control system for interactive physical environments, such as rooms enhanced with multimedia equipment [114]. Their model is an adaptation of role-based access control (RBAC) to the interactive physical environment domain. Their work has a focus on usability in that it is intended to foster collaboration between users, but they still call for administrators to maintain policies in their model.

Agrawal et al. criticize the preference language APPEL, which was designed to allow Web users to specify their preferences for the privacy practices of websites they interact with. Agrawal et al. point out some drawbacks to the APPEL language that cause usability problems for users trying to specify their preferences. Agrawal et al. then go on to suggest a solution that allows Web users to specify privacy preference in the XPath language. However, Agrawal et al. do no usability testing to evaluate their solution [3].

Works addressing the general access-control conflict-resolution methods we have addressed here include Fundulaki and Marx [51], Jajodia et al. [62], Fisler et al. [50], the XACML 1.0 standard [53], Dougherty et al. [43], and Goldberg et al. [54]. Fundulaki and Marx describe an access-control model for eXtensible Markup Language (XML) documents. They acknowledge three conflict-resolution methods: priority precedence, in which a policy author specifies a priority for each rule; deny precedence (which they call "deny overwrites"); and allow precedence (which they call "grant overwrites"). Their semantics uses deny precedence. Jajodia et al. describe an access-control policy framework that accommodates a number of conflict-resolution methods, including specificity, deny, and allow precedence. Their framework also accommodates "no conflicts allowed," in which policies are statically checked for conflicts and conflicts are flagged as errors. Fisler et al. describe Margrave, an RBAC policy-analysis tool, that supports the three conflict-resolution methods outlined in XACML standard: allow precedence (called "permit-overrides" in the standard), deny precedence (called "deny-overrides" in the standard), and order precedence (called "first-applicable" in the standard). The standard can also be extended to accommodate application-specific conflict-resolution methods. Dougherty et al. are concerned with proving properties of safety and availability in access-control policies, comparing policies to each other, and understanding the effects of dynamic changes to the policy environment. They discuss "combiners," the term used in the XACML standard for conflict-resolution methods, as a factor contributing to policy and policy-authoring complexity. Goldberg et al. describe a module they built for restricting access to system calls by untrusted browser helper applications, like document viewers. They use specificity precedence as the conflict-resolution method in their module.

Several works present policy analysis tools that detect conflicts with the goal of helping policy authors resolve them. Al-Shaer and Hamed describe the Firewall Policy Advisor, a tool that detects a variety of firewall policy anomalies, including conflicts [6]. They state that firewall policies traditionally use order precedence to resolve rule conflicts. The Firewall Policy Advisor presents a list-of-rules user interface for analyzing policy conflicts. Yuan et al. present FIREMAN, a toolkit for performing static analysis of firewall policies to check for misconfigurations [148]. Jaeger et al. are concerned not with rule conflicts but with constraint conflicts, which they describe as conflicts in constraints that specify inviolable properties of policies [61]. They present the Gokyo policy analysis tool for constraint conflict detection, analysis, and resolution.

7.4 Policy Semantics Study #1: Do semantics affect usability?

To determine whether policy semantics has an effect on usability, we conducted an extension of the user study described in Chapter 6. The extension study, hereafter referred to as Policy Semantics Study #1, employed the same method as the original, except for the interface condition tested. To isolate the effects of semantics, we implemented an Expandable Grid interface identical to that described in Chapter 6, but using the Windows semantics. (We were able to implement the Grid-with-Windows-semantics condition because our implementation of the file-permissions Expandable Grid interface includes a simulated Windows file access-control system. The simulated system serves as the backend to the Grid interface. Since we wrote the simulated system, we are able to control its semantics.) We compared the performance of 18 participants using the Grid-with-Windows-semantics interface with the performance we observed amongst participants using the Grid (with the recency semantics described in Section 6.1.1) interface in the original study.

We added a dialog box, shown in Figure 7.3, with "Allow" and "Deny" checkboxes to the Grid interface

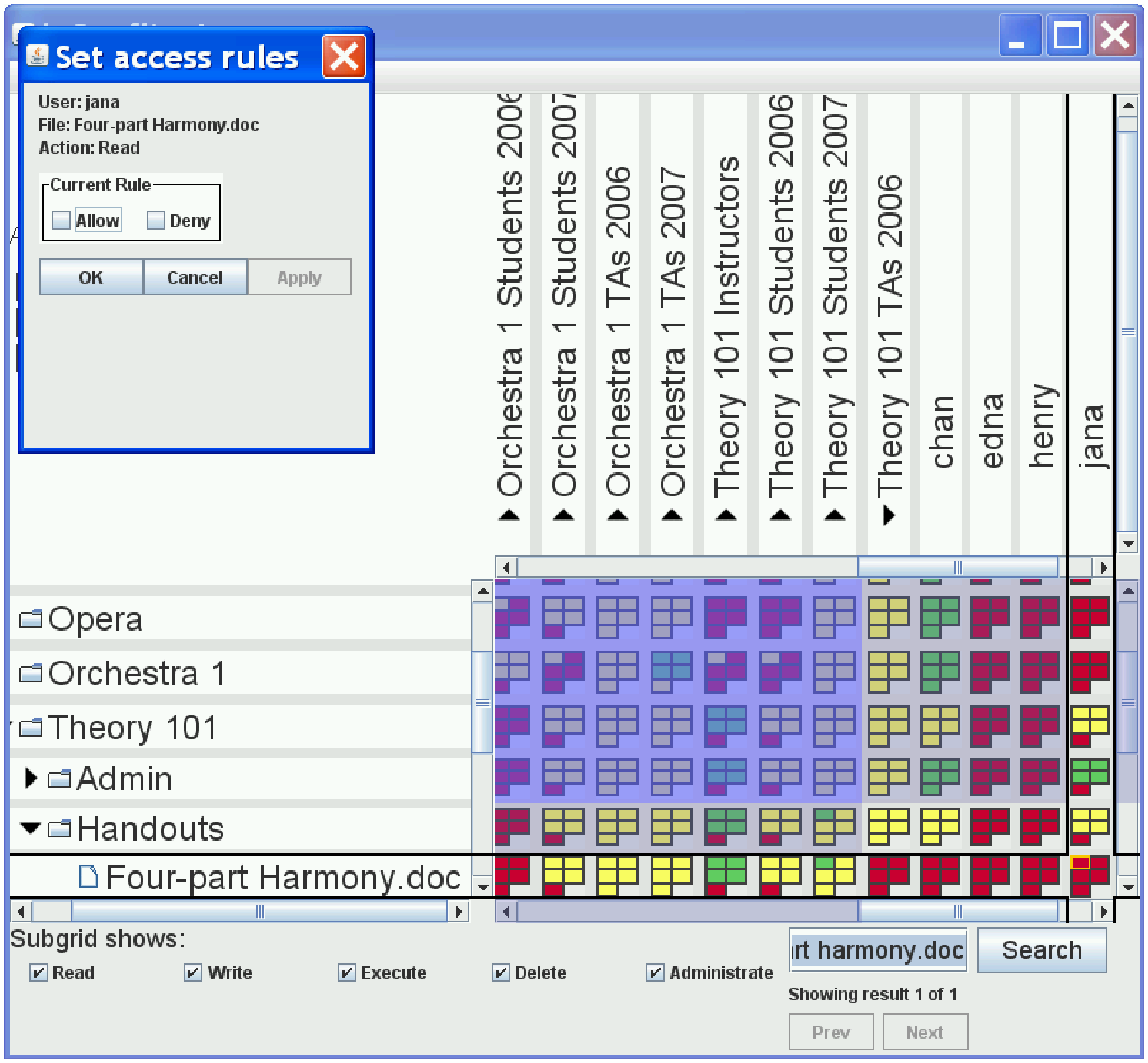


Figure 7.3: *Screenshot of the Grid-with-Windows-semantics interface used in Policy Semantics Study #1. Since direct manipulation is not possible with the Windows semantics, the interface shows a dialog box to allow a policy author to set an access rule.*

to allow authors to set explicit access rules. This dialog box, which appears when the author clicks on a subgrid square, enables the Grid to work with the Windows semantics. The Windows semantics makes it impossible to implement a Grid interface in which clicking a red square always turns it green, i.e., the Windows semantics violates direct manipulation. Because of this characteristic of the Windows semantics, it no longer made sense to hide access rules from authors as we did in the design of the original Grid interface. Moreover, in the Windows semantics, there are three possible access-rule states for each request: ALLOW, DENY, and blank (i.e., no rule set). In the original Grid design, a click on a square only allowed for toggling between two states. The dialog box makes the three states explicit. Note that the dialog box does not allow the invalid state in which both the "Allow" and "Deny" checkboxes are checked. The Grid-with-Windows-semantics still shows effective permissions and shows no explicit representation of the access rules except through the dialog box.

7.4.1 Method

The experimental setup, task design, and procedure for Policy Semantics Study #1 were identical to those of the original study (see Section 6.2). We also recruited 18 participants, nine female, from the same demographic pool (undergraduates in technical majors) and using the same methods (fliers and a local university study-recruitment website) as in the original study, but since we conducted it roughly four months later than the original, it is possible our participant pool had changed somewhat.

7.4.2 Results and discussion

Complete accuracy and time-to-task-completion results for Policy Semantics Study #1, shown with the results from the original study, can be seen in Figures 7.4, 7.5, 7.6, and 7.7. The objective of the study was to determine whether the choice of semantics can have an effect on usability, so we were primarily interested in the results from the tasks in which the semantics comes into play: the change-complex, conflict-complex, and precedence tasks. Note that the semantics does not come into play in any task without a rule conflict or an exception rule. Also, note that the semantics does not come into play in the conflict-simple tasks, since in those cases, the conflict is a DENY exception to an ALLOW rule, which both semantics handle identically. In this study, we were primarily interested in comparisons between the two Grid conditions and not between Grid conditions and Windows, since the Grid conditions offer a controlled comparison between semantics.

If it is true that semantics can affect usability, we would expect to observe differences in performance between the two Grid conditions on the change-complex, conflict-complex, and precedence tasks. Accordingly, we tested for statistically significant differences in accuracy rate and time-to-task-completion between the two Grid conditions on the small- and large-scale change-complex, conflict-complex, and precedence tasks. We used one-sided Fisher's exact tests to test for significant differences in accuracy rates and one-sided t-tests to test to significant differences in times-to-task-completion. We corrected for multiple testing using the Benjamini-Hochberg method, which gave an adjusted α of 0.001 for both accuracy and time-to-task-completion testing. Results of our tests can be seen in Tables 7.2 and 7.3.

From Tables 7.2 and 7.3, we see that four of 12 tests turned out to be statistically significant. Notably, three of the significant tests were on the conflict-complex tasks. This result is not surprising, since the conflict-complex tasks represent precisely the case for which we designed the Grid's recency semantics—the large-scale conflict-complex task is the Jana task we have used as an example of where the Windows

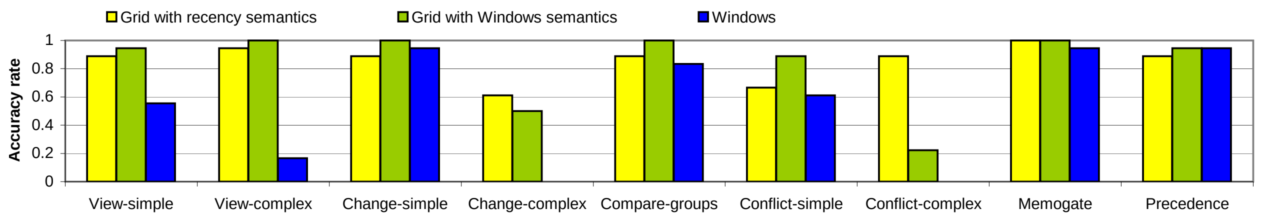


Figure 7.4: *Accuracy results for small-scale tasks for Policy Semantics Study #1, showing proportion of participants correctly completing each task using original Grid (with recency semantics), Grid-with-Windows-semantics, and Windows interfaces. Results for original Grid and Windows interfaces are the same as those presented in Chapter 6.*

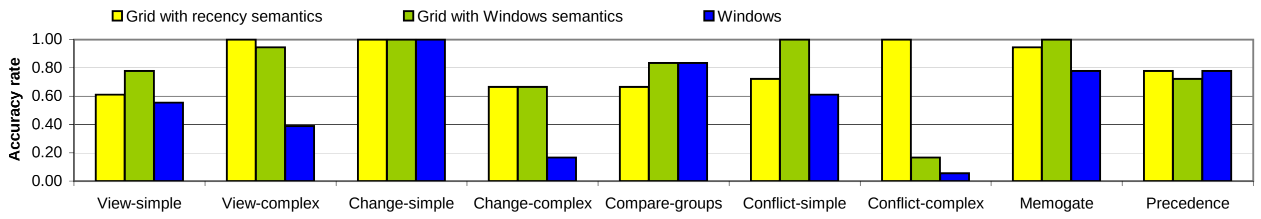


Figure 7.5: *Accuracy results for large-scale tasks for Policy Semantics Study #1, showing proportion of participants correctly completing each task using original Grid (with recency semantics), Grid-with-Windows-semantics, and Windows interfaces. Results for original Grid and Windows interfaces are the same as those presented in Chapter 6.*

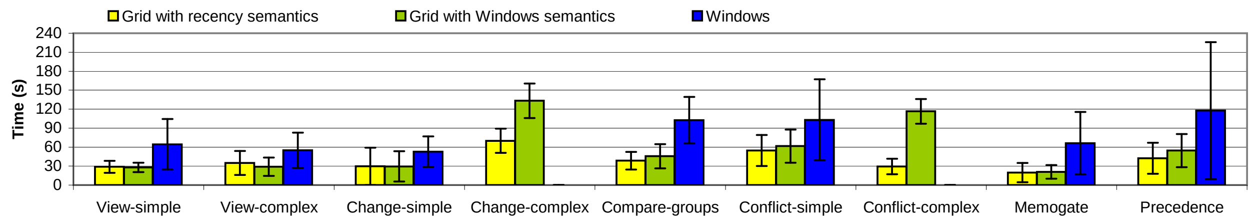


Figure 7.6: *Time-to-task-completion results for small-scale tasks for Policy Semantics Study #1, showing mean time-to-task-completion, in seconds, for participants who correctly completed each task using original Grid (with recency semantics), Grid-with-Windows-semantics, and Windows interfaces. Results for original Grid and Windows interfaces are the same as those presented in Chapter 6. Error bars show +/- one standard deviation.*

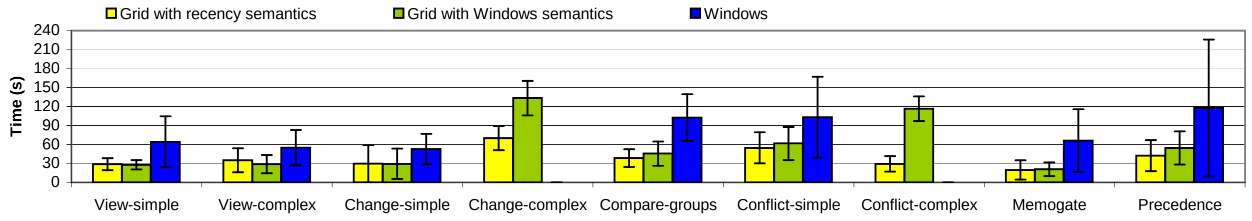


Figure 7.7: *Time-to-task-completion results for large-scale tasks for Policy Semantics Study #1, showing mean time-to-task-completion, in seconds, for participants who correctly completed each task using original Grid (with recency semantics), Grid-with-Windows-semantics, and Windows interfaces. Results for original Grid and Windows interfaces are the same as those presented in Chapter 6. Error bars show +/- one standard deviation.*

Table 7.2: *Summary of statistical tests for significant differences in accuracy rate for Grid with recency semantics (a_{GRS}) and Grid with Windows semantics (a_{GWS}) for each task. For all tests except precedence task tests, the hypothesis tested was $a_{GRS} > a_{GWS}$. For the precedence tests, the hypothesis tested was $a_{GRS} < a_{GWS}$. The p-values shown are from one-sided Fisher's exact tests; p-values below the $\alpha = 0.001$ rejection threshold are shaded and highlighted in bold, indicating significant tests.*

Task pair	Small-scale			Large-scale		
	a_{GRS}	a_{GWS}	p-value	a_{GRS}	a_{GWS}	p-value
Change-complex	0.61	0.50	0.43	0.67	0.67	0.64
Conflict-complex	0.89	0.22	$< \mathbf{0.001}$	1.00	0.17	$< \mathbf{0.001}$
Precedence	0.89	0.94	0.50	0.78	0.72	0.78

Table 7.3: *Summary of statistical tests for significant differences in mean time-to-task-completion, in seconds, between successful Grid with recency semantics (GRS) and Grid with Windows semantics (GWS) participants for each task. For each test, the table shows means (M) and standard deviations (σ) for each interface, and t-statistics (with degrees of freedom (df)) and p-values resulting from one-sided t-tests; p-values at or below the $\alpha = 0.001$ rejection threshold are shaded and highlighted in bold, indicating significant tests.*

Task pair	Small-scale			Large-scale		
	M(σ)	t(df)	p	M(σ)	t(df)	p

Change-complex	GRS:70(19)
	GWS: 133(27)
		−5.44(17)	$< \mathbf{0.001}$	GRS:100(35)
				GWS:110(50)
					−0.01(17)	0.49

Conflict-complex	GRS:29(12) GWS: 116(19)	−10.89(13)	$< \mathbf{0.001}$	GRS:52(37) GWS: 136(93)	−2.51(3)	0.05
Precedence	GRS:42(25) GWS: 55(26)	−1.87(29)	0.96	GRS:71(29) GWS: 88(30)	−1.67(25)	0.95

semantics causes usability problems. In our result for the conflict-complex tasks, we see strong evidence supporting the hypothesis that semantics affect usability.

We did not observe statistically significant differences in the change-complex tasks. We expect the Grid as a presentation technique helps improve authoring performance, even in the presence of an ill-fitted semantics. We did see a significant difference in the times-to-task-completion for the small-scale change-complex task that was likely due to the awkwardness of making policy changes using the Window semantics.

We did not observe statistically significant differences in either accuracy rates or times-to-task-completion for the precedence tasks. This result was not surprising in light of the similar result we saw in Chapter 6, in which there was little difference between the Grid and Windows interfaces on the precedence tasks. As we discussed in Chapter 6, the Grid's policy visualization may mitigate the disadvantage of the recency semantics on the precedence tasks. We also note that the precedence task design is not entirely realistic. In a real policy-authoring scenario, an author would not be explicitly prompted to preserve an exception to a new group access rule, but would have to remember their (or another author's) intent to preserve an exception that was set potentially weeks or months before.

We only compared 12 of 36 measurements we took between the Grid with recency semantics and the Grid with Windows semantics because we were only interested in analyzing tasks in which the semantics came into play. Nevertheless, it is worth looking briefly at the other results to be sure none are awry. As we expect, we see that accuracy rates and times-to-task-completion for the two Grid interfaces are very close to each other. There appears to be a possible performance gain for the Grid with Windows semantics on the conflict-simple tasks. In fact, if we were to test the difference in accuracy rates for the large-scale conflict-simple task for statistical significance, we would achieve a nearly significant result ($p = 0.05$). We caution against doing so, however, as there are bound to be some apparently (but not actually) significant results amongst the many possible significance tests that one could perform. We attribute the apparent performance advantage of the Grid with Windows semantics in the conflict-simple tasks to random variation in our participant pool, not to the semantics.

In summary, we conclude that the policy semantics *does* affect usability. We base this conclusion on the results from tasks involving rule conflicts, for which the recency semantics led to higher accuracy rates compared to the Windows semantics.

7.5 Policy Semantics Study #2: Evaluating our proposed semantics

The strong evidence from Policy Semantics Study #1 suggesting that semantics can affect usability led us to design our specificity semantics. We then ran a second user study to evaluate our new semantics and the concomitant new features of our Grid interface.

7.5.1 Method

We ran a laboratory user study with 54 participants, who each performed 12 policy-authoring tasks using one of three interfaces. Interface was a between-participants variable, so 18 participants used each interface. We measured accuracy and time-to-task-completion for each task. The three interface conditions we compared were:

1. the Grid interface with our specificity semantics, which we henceforth call *GS*;
2. the Grid interface with Windows semantics, which we henceforth call *GW* (note that GW is different from GWS because of the new features we added to the Grid, as discussed in Section 7.2.2); and
3. the Windows interface with the Windows semantics, which we henceforth call *WW*.

The WW interface served as the control in our study; it was the condition on which we hoped to improve. The GS interface is our best effort at improving upon WW, because GS has both the visual presentation advantages of the Expandable Grid and the expected advantages of our specificity semantics. The GW interface serves to help us separate the effects of the Grid as a presentation technique from the effects of the semantics. When we observe differences between the GW and WW conditions, we attribute them to the Grid presentation, and when we observe differences between the GW and GS conditions, we attribute them to the semantics. Differences between the GS and WW conditions are the cumulative effect of the Grid presentation and the specificity semantics.

As in Policy Semantics Study #1, we were able to implement the two different Grid conditions because our implementation of the file-permissions Expandable Grid interface includes a simulated Windows file access-control system. The simulated system serves as the backend to the Grid interface. Since we wrote the simulated system, we are able to control its semantics. We implemented both the Windows semantics and our specificity semantics in the simulated system, so the Grid interface can run on top of either semantics.

7.5.1.1 Participants

We recruited 54 students, undergraduate and graduate, in technical disciplines (science, engineering, or mathematics) to participate in the study. Eighteen were female. Participants were all daily computer users, but had never served as system administrators. Like our participant pool from the study described in Chapter 6, our participant pool was consistent with our target demographic of novice and occasional policy authors. However, the participant pool in the present study did include graduate students (27 of 54 participants were graduate students), so may be a more advanced group than that in the prior study.

7.5.1.2 Experimental setup

Participants worked on a laptop running the Windows XP Professional operating system. Task statements were presented in a Web browser and were available on screen throughout the task. For participants using Windows, we started the Computer Management interface in a state that allowed for viewing system users and groups and opened Windows Help files related to setting file permissions. For participants using the Expandable Grid, the Grid application window was started at an initial size of 800x740 pixels. The laptop screen resolution was 1280x768.

The data we collected included the policies people created and screen video and audio of participants thinking aloud.

7.5.1.3 Task design

We designed 12 tasks to test the advantages and disadvantages of each of the two semantics as well as the overall usability of each interface. All tasks were based on the same teaching-assistant scenario used in the study described in Chapter 6, but tasks in this study used only "large-scale" policies (containing roughly 500 principals and 500 resources). Each task is defined by its task statement (i.e., the text we presented to participants in the study) and its initial configuration, including existing access rules, group memberships, and file locations. Task statements asked participants to make changes to the initial configuration. We describe each of the tasks below.

Charles

The Charles task involves adding a user to a group, and that user has several ALLOW permissions on some files, the group has DENY permissions on those files, and the goal is to keep the user's ALLOW permissions.

The Charles task statement presented to participants was:

> Charles has just graduated, but is going to come back to sing in the choir with his friends.
>
> Add **Charles** to the **Alumni** group, but make sure he can still **read** the same files in the **Choir 1\Lyrics** folder that his good friend Carl can read.

In the initial configuration, there are rules stating that Charles is allowed READ access to four files in the Choir 1\Lyrics folder. These are the same files that Carl can read, so in the final state, we want Charles to be allowed READ access to the four files. There are rules stating that the Alumni group is denied READ access to the same files. When Charles is moved into the Alumni group, he will inherit the group DENY rules, and under the Windows semantics deny-precedence, he will be denied access to the files. However, under specificity semantics, his rules are more specific than the group rules, so he will still be allowed to read the files (see Figure 7.2). Thus, the specificity semantics makes this task easier, and we expect the GS interface to perform better than GW and WW in the Charles task.

Kent

The Kent task was structured similarly to the Charles task, but the goal was inverted to give the advantage to the Windows semantics.

The Kent task statement presented to participants was:

> Kent was a terrible TA for Choir 1 so the instructor demoted him to the level of student. While Kent previously had permissions to read and write the attendance and gradebook files, as a student he should no longer have access to that information.
>
> Remove **Kent** from **Choir 1 TAs 2008** and add him to **Choir 1 Students 2008**. For files in the **Classes\Choir 1\Admin** folder, make sure he only has the same permissions as the other Choir 1 students.

In the initial configuration, there are rules stating that Kent is allowed READ and WRITE access to the attendance.xls and gradebook.xls files in the Classes\Choir 1\Admin folder, but there are rules stating that the group Choir 1 Students 2008 is denied READ and WRITE access to those files. So, when Kent is added to Choir 1 Students 2008, he will inherit the group's DENY rules. Under the Windows semantics, the group DENY rules will take precedence, but under specificity semantics, he will still be allowed to read and write the files. Thus, the task is done after moving Kent out of the TA group and into the student group in the GW and WW conditions, but extra action is required to complete the task in the GS condition. So, we expect the Windows semantics, embodied in GW and WW, to perform better on the Kent task. However, we expect the Grid's visual presentation advantages over the Windows list-of-rules presentation at least partially to offset the semantics disadvantage.

Lance

The Lance task introduces a two-dimensional conflict, in which there are conflicts in both the principals and resources.

The Lance task statement presented to participants was:

> Lance was hired by the New York Philharmonic and can no longer serve as Head TA this year.
>
> Remove **Lance** from the group **Head TAs 2008**, but make sure you don't remove him from **Head TAs 2007**. Then make sure he is not allowed to access any files in the **Music 101\Admin** folder.

In the initial configuration, there is a rule stating that Lance is denied READ access to the Music 101\Admin folder, and there are rules stating that the Head TAs 2007 group is allowed READ access to the Music 101\Admin\gradebook.xls file, but that the Head TAs 2008 group is denied READ access to the file. When Lance is removed from Head TAs 2008, a two-dimensional conflict is revealed, since the rule applying to Lance on the Admin folder is more specific in its principal, but the rule applying to Head TAs 2007 on the gradebook.xls file is more specific in its resource. In the Windows semantics, the rule that is more specific in the resource takes precedence, so Lance will be allowed to read the gradebook.xls file, but in specificity semantics, the DENY rule takes precedence, so Lance will not be allowed to read the gradebook.xls file. Since the task calls for Lance not to be allowed to access any files in the Admin folder, the specificity semantics requires less work to complete the task correctly, and we expect the GS interface to perform best here.

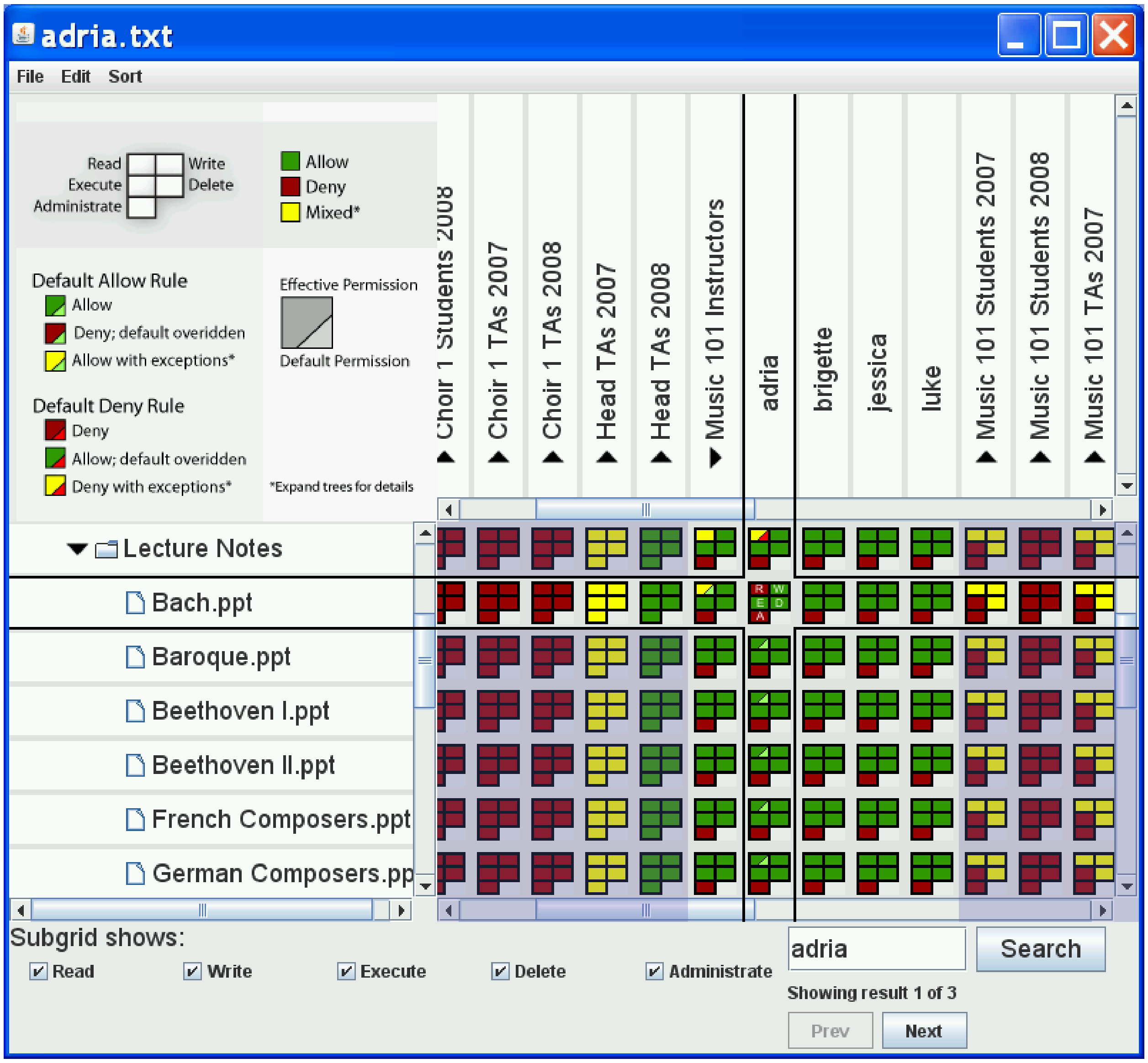
Adria

Just as the Kent task is similar to the Charles task with the inverse goal, the Adria task is similar to the Lance task with the inverse goal.

The Adria task statement presented to participants was:

> Adria, an Opera Instructor, was not getting along with the other instructor and left the class. You need to remove her from the Opera Instructors group. She is still a Music 101 instructor, though, and the Music 101 instructors need access to the Music 101 Lecture Notes.
>
> Remove **Adria** from the **Opera Instructors** group. Make sure she has the same permissions on the files in the **Music 101\Lecture Notes** folder as the other Music 101 instructors.

As in the Lance task, when Adria is removed from a group, a two-dimensional conflict involving another group is revealed. In the initial configuration, there is a rule stating that Adria is denied READ access to the Music 101\Lecture Notes folder and there is a rule stating that the Music 101 Instructors group is allowed READ access to the Music101\Lecture Notes\Bach.ppt file. This latter group access rule is initially suppressed in both semantics by a rule stating that the Opera Instructors group is denied READ access to the Bach.ppt file, but when Adria is removed from Opera Instructors, a two-dimensional conflict is revealed (see Figure 7.8). In the Windows semantics, the group access rule, which is more specific in its resource, will take precedence and Adria will be allowed to read the Bach.ppt file as soon as she is removed from Opera Instructors. In our specificity semantics, deny-precedence is used to resolve two-dimensional conflicts, so Adria will not be allowed to read the Bach.ppt file without some extra work. We thus expect the GW and WW conditions to perform better than the GS condition for the Adria task. However, the extra work required



Subgrid shows:

☑ Read ☑ Write ☑ Execute ☑ Delete ☑ Administrate

adria

Showing result 1 of 3

Figure 7.8: *A screenshot of the Grid interface from the Adria task after Adria has been removed from the Opera Instructors group. There is a two-dimensional conflict for Adria's* READ *permission for the Bach.ppt file.*

in the GS condition is quite simple. With the specificity semantics, simply setting an access rule explicitly allowing Adria to read the Bach.ppt file is sufficient, as the rule applying to Adria and Bach.ppt will be more specific than the rules in conflict. We thus expect that the GS's presentation advantages over the WW condition combined with the ease of overcoming the two-dimensional conflict might counteract the WW semantics advantage.

Pablo

The Pablo task, like the Jana task described in Chapter 6 presents a group conflict in which a user is a member of two groups, one of which has a rule allowing it access to a file and the other of which has a rule denying it access to the same file.

The Pablo task statement presented to participants was:

> Pablo, a student in Music 101, tried to download the homework file, assignment4.pdf, but couldn't.
>
> Set permissions so that **Pablo** can **read** the file **assignment4.pdf** in the **Music 101\Handouts** folder. Make sure you don't change any other students' permissions. (Hint: If you need to, you can add Pablo to a new group or remove Pablo from a group he's already in.)

In the initial configuration, Pablo is a member of the group Music 101 Students 2008, for which there is a rule allowing the group READ access to the assignment4.pdf file, and he is a member of the group Troublemakers, for which there is a rule denying the group READ access to the assignment4.pdf file. In both semantics, deny-precedence is in effect for conflicts between groups. The difference between the semantics comes into play when participants try to resolve the conflict. In the specificity semantics, authors can simply create a rule allowing Pablo to read assignment4.pdf, and it will take precedence because it will be more specific than either of the group rules. In the GS interface, creating such a rule is a simple matter of clicking on the square corresponding to Pablo, assignment4.pdf, and READ. In the Windows semantics, such a rule would not take precedence; completing the task requires either removing Pablo from the Troublemakers group or removing the rule that denies Troublemakers READ access to the file. In either solution, care must be taken not to change the effective permissions of other members of the Troublemakers group. Because there are more steps involved in the solution for the Windows semantics, we expect better performance with the GS interface for the Pablo task.

Piano

The Piano task tests the interfaces' group-creation functionality and introduces a group conflict.

The Piano task statement presented to participants was:

> A new seminar on piano performance was just started.
>
> Create a group called **Piano Students 2008** and add the following students to it: **Aaron, Camilla, Thor, and Uzi**. Set permissions so that these students and any students added to Piano Students 2008 in the future will be able to **read** the **Piano\Handouts** folder.

We expect creating the group, adding users to it, and giving it READ access to the Handouts folder to be easy, but there is a trick in our configuration. The user Thor is a member of the Troublemakers group, for which there is a rule that denies it READ access to the Handouts folder. In both semantics, the DENY rule takes precedence in the presence of the group conflict. However, as in the Pablo and Jana tasks, the conflict is much more easily resolved with the specificity semantics, because resolving the conflict is a simple matter of creating a rule stating that Thor is allowed READ access to the Handouts folder. With the Windows semantics, it is necessary to either remove Thor from the Troublemakers group or remove the rule denying Troublemakers access to the Handouts folder. We expect the GS interface to perform best for the Piano task.

In addition, we expect the GW interface to perform better than the WW interface. Correctly completing the task requires noticing that Thor does not have READ access to the Handouts folder. We expect it to be easier to read Thor's effective permissions with the Grid's full, effective policy presentation than with the Windows list-of-rules presentation, in which effective policy is three mouse clicks away from the main interface display.

Troublemakers

The Troublemakers task tests the ability of the interfaces to reveal an undesired exception to a rule.

The Troublemakers task statement presented to participants was:

> The music department is full of pranksters. These people have been put in the Troublemakers group.
>
> Set permissions so that **no one** in the **Troublemakers** group has access to anything.

In the initial configuration, there are rules denying all access to the root Classes folder to members of the Troublemakers group, but there is one user who has access to a folder two levels below the root folder. The Windows interface only allows the author to check the effective policy for one user for one resource at a time. In our configuration, there are seven users in the Troublemakers group and 29 folders one or two levels below the root, so it could take up to 7x29=203 operations to ensure that no members of the Troublemakers group have access to any of those folders using the Windows interface. The Grid interfaces show the aggregate effective policy for READ and WRITE for the Troublemakers at the root folder as mixed (yellow squares with red dog-ears), thereby providing the author a clue that some member of Troublemakers has access to something. Expanding the group reveals that Marie is the user with access to something within the Opera folder, and expanding the Opera folder reveals two rules stating that Marie has access to read and write the Opera\Admin folder. See Figure 7.9. Because of the relative ease of spotting exceptions to the desired DENY rule in the Grid interfaces, we expect them to perform better in the Troublemakers task compared to the Windows interface. The semantics does not play a role in the Troublemakers task.

Assignment

The assignment task tests the semantics in the presence of file moves and illustrates a potentially confusing aspect of the Windows semantics.

The Assignment task statement presented to participants was:

> The Music 101 Assignment 1 had some mistakes and needs to be edited. Students should be able to read the file while it is being edited, and TAs will need to read and write the file.
>
> Move the **assignment1.pdf** file from the **Classes\Music 101\Handouts** folder to the **Classes\Music 101\Drafts** folder. Then set permissions so that:
>
> - **Music 101 Students 2008** can **read** the **assignment1.pdf** file; and

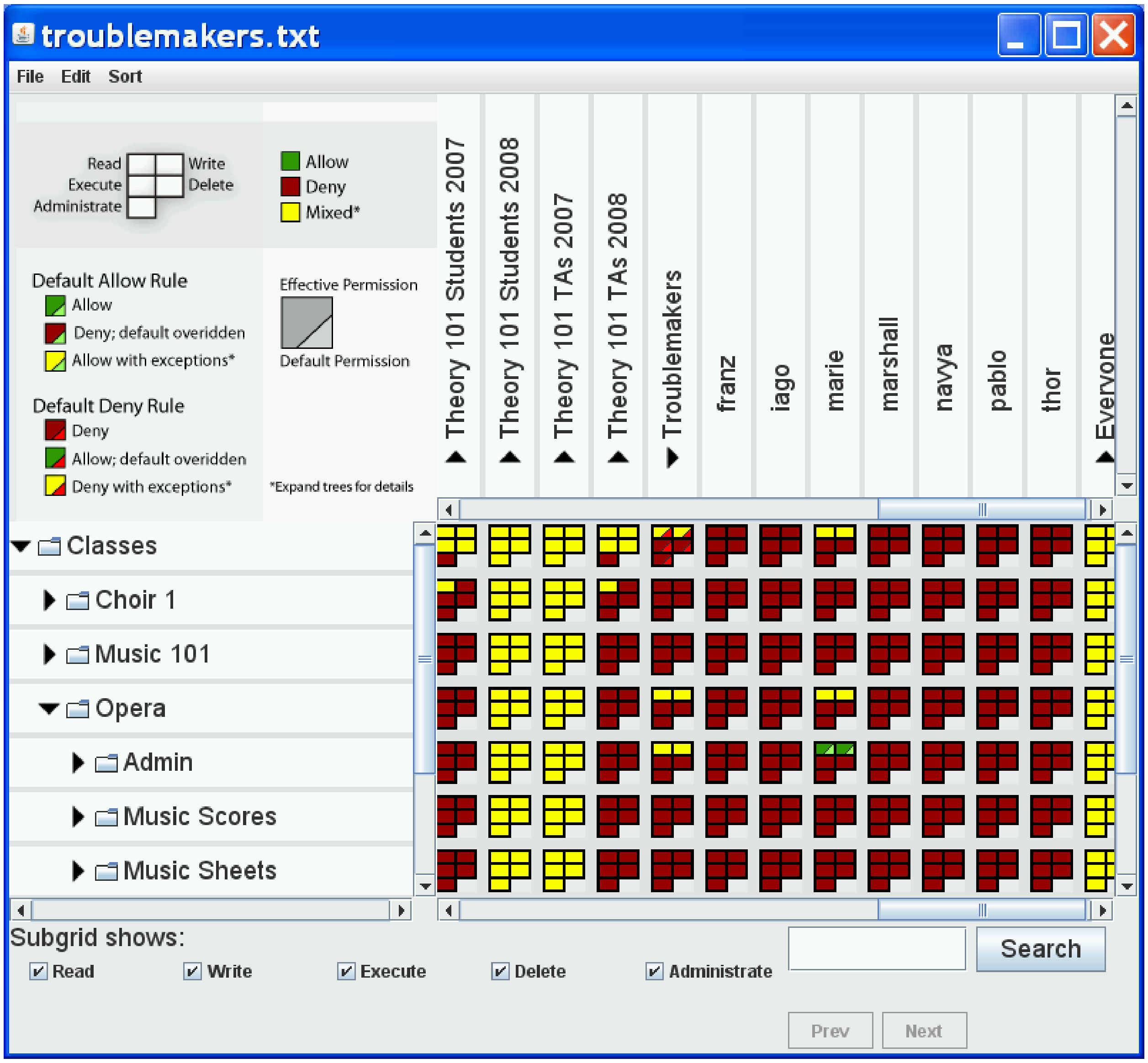


Figure 7.9: *A screenshot of the Grid interface for the Troublemakers task, showing the user "marie" with an exception to the group's* DENY *rule on the root Classes folder.*

- **Music 101 TAs 2008** can **read and write** the **assignment1.pdf** file.

In the Windows semantics, when a file is moved from one parent folder to another, it retains the rules it inherited from its old parent. When a change is made to the file's or its new parent's access control list, the file then inherits all rules from the new parent, and loses rules from the old parent. In our specificity semantics, a moved file inherits rules from its new parent immediately.

In the initial configuration for the Assignment task, relevant rules state that:

- Music 101 Students 2008 are allowed READ access to files in the Handouts folder;
- Music 101 Students 2008 are denied READ access to files in the Drafts folder;
- Music 101 TAs 2008 are allowed READ access but denied WRITE access to files in the Handouts folder; and
- Music 101 TAs 2008 are allowed READ and WRITE access to files in the Drafts folder.

From the rules, we see that if the assignment1.pdf file is moved and inherits rules from its new parent folder Drafts, the task will be correctly completed. Thus, under our semantics, the Assignment task is complete as soon as the participant has moved the file. Under the Windows semantics, though, participants may be tricked by the rule that the file retains its old parent's rules, but may suddenly inherit all of its new parent's rules if the participant makes a change. So, under the Windows semantics, after the participant has moved the assignment1.pdf file, they may check to see that Music 101 Students 2008 can read the file. Because students are allowed to read files in the old parent folder Handouts, the participant will note that the students have the correct access. The participant may then proceed to check what access the Music 101 TAs 2008 have to the assignment1.pdf file, and note that they have can read but not write the file. However, when the participant creates a rule allowing the TAs WRITE access to the file, the file will inherit all rules from its new parent, including the rule that the students are denied READ access to files in the Drafts folder. Unless the participant goes back and checks the students' access again, they will end up with the wrong access and the task will not be completed correctly. Thus, we expect that the GS interface will perform best in the Assignment task.

Syllabus

The syllabus task is structured similarly to the Assignment task but with the inverse goal, so it gives the advantage to the Window semantics.

The Syllabus task statement presented to participants was:

> The Music 101 syllabus was in draft form, but is now complete and ready for students to read.
>
> Move the **syllabus.doc** file from the **Classes\Music 101\Admin** folder to the **Classes\Music 101\Handouts** folder. Then set permissions so that all members of **Music 101 Students 2008** can **read** the **syllabus.doc** file.

In the initial configuration, there is a rule stating that all students have READ access to the syllabus.doc file, but there are rules stating that four of the students are denied READ access to files in the Handouts folder. Under the Windows semantics, moving the file does not change the students' access, and since they all already have READ access to the file, the task is complete as soon as the participant moves the file into the Handouts folder. In our semantics, when the participant moves the file, it inherits its new parents' rules, so the rules denying READ access to the four students go into effect. The participant must make the extra effort to check the effective policy for those students and explicitly grant them READ access to correctly complete the task. Thus, we expect the GW and WW interfaces with the Windows semantics to perform better in the Syllabus task.

Jana

The Jana task was described in Chapter 6, and, as mentioned above, is similar in structure to the Pablo task.

The Jana task statement presented to participants was:

> Jana, a Theory 101 TA, complained that when she tried to change the Four-part Harmony handout to update the assignment, she was denied access.
>
> Set permissions so that **Jana** can **read and write** the **Four-part Harmony.doc** file in the Theory 101\Handouts folder.

As explained in the description of the Pablo task above, we expect specificity semantics to perform better for the Jana task than the Windows semantics. Moreover, we have already seen in Chapter 6 that a recency semantics, which is equivalent to our specificity semantics as far as the Jana task is concerned, outperformed the Windows semantics. We expect the specificity semantics in the GS interface to outperform the GW and WW interfaces with the Windows semantics for the Jana task.

Clayton

The Clayton task is the large-scale Conflict-simple task from the study described in Chapter 6.

The Clayton task statement presented to participants was:

> Clayton, a Theory 101 TA, is going away on a trip and cannot grade the Simple Solo assignment. Because of this he will be put in charge of grading the Simple Harmony assignment. Clayton will need read and write access to the Simple Harmony assignment. He also may wish to refer to the Simple Solo assignment, so he should be allowed to read the Simple Solo submissions folder. However, you don't want him to accidentally overwrite the submissions to Simple Solo when viewing them.
>
> Set permissions so that **Clayton** can **read and write** the **Simple Harmony** subfolder in the Theory 101\Submissions folder.
>
> Set permissions so that **Clayton** can **read, but not write**, the **Simple Solo** subfolder in the Theory 101\Submissions folder.

We observed little difference between the GS and WW interfaces in the study reported in Chapter 6, and we expect little difference here.

Quincy

The Quincy task is the large-scale Precedence task from the study described in Chapter 6.

The Quincy task statement presented to participants was:

> Students in the Choir class are going to sing the War Requiem, so the Choir instructor, Savanna, has asked you to give them access to the War Requiem lyric sheets. However, you remember

> that last week she asked you to prevent Quincy from accessing any tenor parts, because, although he obnoxiously insists he is a tenor, Savanna wants him to sing baritone. You asked, and she confirmed that Quincy should not have access to the tenor part. Other students should have access to all the parts.
>
> Set permissions so that **Choir 1 Students 2008** can **read** the files in the **War Requiem** subfolder of the Choir 1\Lyrics folder. But remember that Quincy should not be allowed to read the tenor part.

We designed the Quincy task to test a weakness of our recency semantics. In a recency semantics, a rule at the group and folder level can override an exception at the user and file level that was intended to be kept. In the initial configuration for the Quincy task, there is rule denying Quincy READ access to the Tenor.pdf file in the War Requiem folder. Under a recency semantics, when a rule is created to allow all students READ access to all files in the War Requiem folder, the DENY rule applying to Quincy will be overridden. However, under both the Windows semantics and our specificity semantics, the exception will stay in place because it is more specific than the new rule applying to students and the War Requiem folder. Thus, we expect similar performance amongst the interfaces for the Quincy task.

7.5.1.4 Procedure

At the start of each study session, participants filled out a demographic survey so that we could ensure they were students in technical disciplines. Following the survey, our experimenter read instructions explaining our teaching-assistant scenario to participants. After reading these instructions, our experimenter read interface training materials. Training for this study was more extensive than that for Policy Semantics Study #1 and the original study described in Chapter 6, since performing dynamic changes required participants to understand more functionality. For each interface, training covered how to perform the following operations:

- viewing files and folders;
- moving a file;
- viewing group memberships;
- adding a user to a group;
- removing a user from a group;
- creating a new group;
- checking an access rule;
- checking effective permissions;
- creating an access rule;
- searching for a file or principal.

During training, the experimenter also explained that effective permissions may differ from access rules because of the way rules combine, but did not explain the precise workings of the semantics. After these operations had been explained to participants, the experimenter walked them through a practice task. Training took about 10 minutes.

Participants then began completing tasks. Before each task, the experimenter brought up the interface in a preconfigured state tailored to each task. Task statements were then presented to participants in a Web browser on screen. Participants were asked to think aloud while they worked on the tasks. Task order was counterbalanced across participants using a pseudo-random Latin square design to guard against ordering and sequence effects.

7.5.2 Results and discussion

Our results are in the form of accuracy rates and mean time-to-task-completion for each interface condition and for each task. Accuracy rates represent the proportion of participants in each condition who correctly completed the task. The mean time-to-task-completion is computed only over the task completion times of those participants who successfully completed each task. Accuracy results can be seen in Figure 7.10. Time-to-task-completion results can be seen in Figure 7.11.

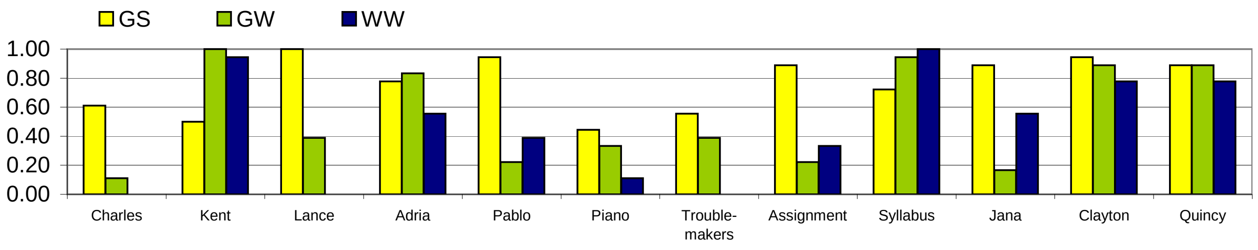


Figure 7.10: *Accuracy results, showing proportion of participants correctly completing each task with GS, GW, and WW interfaces.*

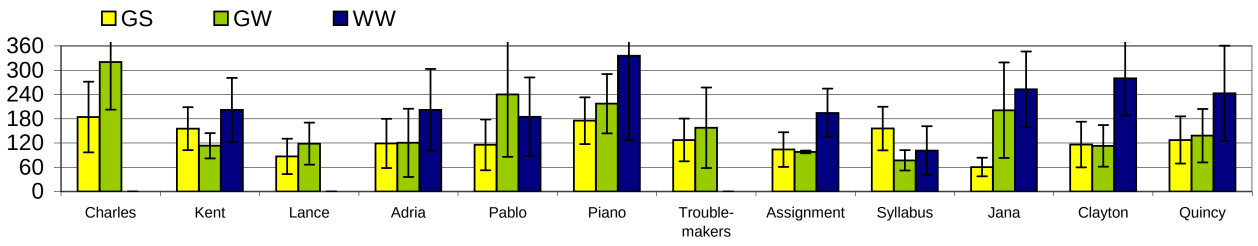


Figure 7.11: *Time-to-task-completion results, showing mean time-to-task-completion, in seconds, for participants who successfully completed each task with GS, GW, and WW interfaces. Error bars show +/- one standard deviation.*

Experiment-wide results

For each metric, accuracy and time-to-task-completion, we performed an experiment-wide test of the hypothesis that GS performed better than GW and WW and that GW performed better than WW over all tasks. The accuracy rates over all tasks do show that GS performed best (overall accuracy 76.4%), followed by GW (overall accuracy 53.2%) and then WW (overall accuracy 45.4%). We used logistic regression to test for statistical significance of the effects of interface on accuracy. The fitted model gave an intercept of 1.17 for the GS condition and gave coefficients of -1.04 for the GW condition and -1.36 for the WW condition. Wald tests of the hypotheses that the intercept ($Z = 7.33, p < 0.001$) and the coefficients ($Z = -4.96, p < 0.001; Z = -6.46, p < 0.001$) were not equal to zero were all strongly significant at the 0.05 level, suggesting that interface has an effect on accuracy.

We used an ANOVA with interface as the independent variable and log-transformed time-to-task- completion as the dependent variable. We found a statistically significant effect of interface (F(2,375)=32.17, p<0.001) on time-to-task-completion. The mean times-to-task-completion (measured in seconds) show GS leading to fastest task completion (M=121, σ=63), followed by GW (M=130, σ=78) and WW (M=207, σ=106).

Both the analysis of overall accuracy and overall time-to-task-completion show the result we expected, which is best performance from GS, medium performance by GW, and worst performance by WW. However, results of the overall analysis should be interpreted with some caution since we did not design the set of tasks to be a balanced representation of all tasks a policy author might do in practice. Instead, we designed each task to test a specific aspect of the specificity semantics or of the Grid interface. Following are the task-by-task results, which address the effects of specific aspects of the semantics and the Grid on policy-authoring performance.

Task-by-task analysis method

We designed our 12 tasks to reveal strengths and weaknesses of the specificity and Windows semantics and the Grid and Windows policy presentations. In Sections 7.5.2.1 through 7.5.2.11, we state our hypotheses for each task and the results of statistical testing of the hypotheses, and discuss the results. A broader discussion of the overall results of the study follows the task-by-task presentation of results.

Our hypotheses for each task express our expectations, based on each task design, about how the accuracy and time-to-task-completion results would turn out. We tested hypotheses regarding accuracy rates by using two-sided Fisher's exact tests with the null hypothesis that the difference in accuracy rates was zero. We tested hypotheses regarding time-to-task-completion by using two-sided t-tests on log-transformed mean times-to-task-completion with the null hypothesis that the difference in mean time-to-task-completion was zero. Although our hypotheses are often directional (i.e., we expect better performance from one condition than another) we used two-sided tests in all cases. The decision to use two-sided tests is safe since two-sided tests are more conservative than one-sided tests.

Because we tested 24 hypotheses regarding accuracy and 19 regarding time-to-task-completion, we applied a multiple testing correction to our $\alpha = 0.05$ significance threshold. We used the Benjamini-Hochberg correction which gave us adjusted α values of 0.007 for the accuracy tests and 0.01 for the time-to-task-completion tests.

We use the notation a_{GS}, a_{GW}, and a_{WW} to represent the accuracy rates for the GS, GW, and WW conditions and ttc_{GS}, ttc_{GW}, and ttc_{WW} to represent mean time-to-task-completion over successful participants for the three conditions.

Results of all hypothesis tests of accuracy rates can be seen in Table 7.4. Results of hypothesis tests of times-to-task-completion can be seen in Table 7.5.

Table 7.4: *Summary of statistical tests for significant differences in accuracy rate for Grid with specificity semantics (a_{GS}), Grid with Windows semantics (a_{GW}), and Windows (a_{WW}). For each task, the table shows accuracy rates for the three interfaces, hypotheses tested, and p-values from two-sided Fisher's exact tests; p-values below the $\alpha = 0.007$ rejection threshold are shaded and highlighted in bold, indicating significant tests.*

Task	a_{GS}	a_{GW}	a_{WW}	hypothesis	p-value
Charles	0.61	0.11	0.00	$a_{GS} > a_{GW}$	**0.004**
				$a_{GS} > a_{WW}$	**0.001**
Kent	0.50	1.00	0.94	$a_{GS} < a_{GW}$	**0.001**
				$a_{GS} \neq a_{WW}$	**0.007**
Lance	1.00	0.39	0.00	$a_{GS} > a_{GW}$	**0.001**
				$a_{GS} > a_{WW}$	**0.001**
				$a_{GW} > a_{WW}$	**0.007**
Adria	0.78	0.83	0.56	$a_{GS} < a_{GW}$	1.00
				$a_{GS} \neq a_{WW}$	0.29
Pablo	0.94	0.22	0.39	$a_{GS} > a_{GW}$	**0.001**
				$a_{GS} > a_{WW}$	**0.001**
				$a_{GW} > a_{WW}$	0.47
Piano	0.44	0.33	0.11	$a_{GS} > a_{GW}$	0.73
				$a_{GS} > a_{WW}$	0.061
				$a_{GW} > a_{WW}$	0.23
Troublemakers	0.56	0.39	0.00	$a_{GS} > a_{WW}$	**0.001**
				$a_{GW} > a_{WW}$	**0.007**
Assignment	0.89	0.22	0.33	$a_{GS} > a_{GW}$	**0.001**
				$a_{GS} > a_{WW}$	**0.002**
Syllabus	0.72	0.94	1.00	$a_{GS} \neq a_{GW}$	0.17
				$a_{GS} \neq a_{WW}$	0.05
Jana	0.89	0.17	0.56	$a_{GS} > a_{GW}$	**0.001**
				$a_{GS} > a_{WW}$	0.06
				$a_{GW} > a_{WW}$	0.035
Clayton	0.94	0.89	0.78	*No test*	
Qunicy	0.89	0.89	0.78	*No test*	

Scoring methods for accuracy

In computing accuracy rates, we identified two possible methods for scoring a participant's solution to a task as correct or incorrect. Participants sometimes made extraneous changes to policies that were unnecessary

Table 7.5: *Summary of statistical tests for significant differences in mean time-to-task-completion, in seconds, between successful participants in GS (ttc_{GS}), GW (ttc_{GW}), and WW (ttc_{WW}) conditions. For each task, the table shows means (M) and standard deviations (σ) for each interface, the hypotheses tested, and t-statistics (with degrees of freedom (df)) and p-values resulting from two-sided t-tests; p-values at or below the $\alpha = 0.01$ rejection threshold are shaded and highlighted in bold, indicating significant tests. "Ins. data" indicates places where insufficient data was available to compute a statistic.*

Task	GS: M(σ)	GW: M(σ)	WW: M(σ)	test	t(df)	p
Charles	184(87)	320(117)	*Ins. data*	$ttc_{GS} < ttc_{GW}$ $ttc_{GS} < ttc_{WW}$	-2.04(1.89) *Ins. data*	0.19
Kent	155(53)	113(31)	202(79)	$ttc_{GS} > ttc_{GW}$ $ttc_{GS} > ttc_{WW}$	2.26(13.2) -1.55(19.2)	0.04 0.14
Lance	87(44)	118(52)	*Ins. data*	$ttc_{GS} < ttc_{GW}$ $ttc_{GS} < ttc_{WW}$ $ttc_{GW} < ttc_{WW}$	-1.35(10.6) *Ins. data* *Ins. data*	0.20
Adria	119(61)	120(84)	202(101)	$ttc_{GS} > ttc_{GW}$ $ttc_{GS} > ttc_{WW}$	0.09(27.0) -2.75(21.2)	0.92 **0.01**
Pablo	115(63)	240(154)	185(97)	$ttc_{GS} < ttc_{GW}$ $ttc_{GS} < ttc_{WW}$ $ttc_{GW} < ttc_{WW}$	-1.88(3.9) -1.69(9.8) 0.57(5.8)	0.14 0.12 0.59
Piano	175(58)	217(73)	336(209)	$ttc_{GS} < ttc_{GW}$ $ttc_{GS} < ttc_{WW}$ $ttc_{GW} < ttc_{WW}$	-1.05(10.0) -1.21(1.1) -0.77(1.2)	0.32 0.42 0.56
Troublemakers	127(53)	158(100)	*Ins. data*	*No test*		
Assignment	103(42)	98(3)	194(60)	$ttc_{GS} < ttc_{GW}$ $ttc_{GS} < ttc_{WW}$	-0.18(15.8) -3.93(11.2)	0.86 **0.002**

Syllabus	156(54)	77(25)	101(60)	$ttc_{GS} \neq ttc_{GW}$
				$ttc_{GS} \neq ttc_{WW}$
					4.97(26.0)
					3.20(28.0)
						0.001
						0.003

Jana	60(23)	201(118)	253(94)	$ttc_{GS} < ttc_{GW}$ $ttc_{GS} < ttc_{WW}$ $ttc_{GW} < ttc_{WW}$	-2.40(2.1) -10.1(19.2) -0.74(2.3)	0.13 **0.001** 0.53
Clayton	116(56)	113(52)	280(92)	*No test*		
Quincy	127(58)	138(66)	243(118)	*No test*		

to complete the tasks. For example, some participants in the Adria task changed the DENY rule for Adria's READ access to the entire Music 1\Lecture Notes folder to an ALLOW rule. Doing so accomplishes the task's explicit goal of allowing Adria to read the Bach.ppt file. However, changing the folder-level rule could have undesirable side effects, since it affects Adria's access to other files in the folder. So, while the task statement did not explicitly prohibit changing Adria's permissions on the folder, changing these permissions could have undesirable consequences in a real scenario.

One of the two scoring methods scores a solution as correct as long as the final effective policy fulfills all explicitly stated goals of a task, regardless of any extraneous policy changes. Under this method, the solution of changing Adria's DENY rule on the Lecture Notes folder was scored as correct. A stricter scoring method scores a solution as correct if it not only fulfills the explicit goals of the task, but also does not change any other parts of the effective policy. Under this method, changing Adria's folder-level rule was scored as incorrect.

In the results reported here, we used the first, less strict scoring method for consistency with the method we used in the study in Chapter 6. The proportions by which accuracy would change using the other scoring method are shown in Table 7.6. We note that extraneous policy changes were relatively rare, and using a different scoring method would make little difference in our results. Applying the stricter scoring method would change the statistical significance of only three results: (1) the accuracy difference between GS and WW for the Kent task would not be statistically significant; (2) the accuracy difference between GS and WW for the Pablo task would not be statistically significant; and (3) the difference between GS and WW for the Jana task would be statistically significant.

Table 7.6: *Proportion of otherwise-correct solutions with extraneous changes by interface condition and task. If we were to use a stricter accuracy scoring method, we would subtract the proportions in columns 2-4 from the corresponding accuracy rates in Table 7.4.*

	Proportion of solutions with extraneous changes		
Task	GS	GW	WW
Charles	0.06	0.06	0.00
Kent	0.00	0.11	0.17
Lance	0.06	0.00	0.00
Adria	0.17	0.00	0.17
Pablo	0.22	0.00	0.00
Piano	0.00	0.06	0.00
Troublemakers	0.00	0.00	0.00
Assignment	0.00	0.00	0.06
Syllabus	0.00	0.00	0.11
Jana	0.00	0.06	0.11
Clayton	0.17	0.17	0.39
Quincy	0.00	0.06	0.22

Error analysis

To gain more insight into the reasons why accuracy rates turned out as they did, we performed an error analysis for each task. After watching several participants complete the study, we developed categories of error types. Then, for each incorrectly completed task in the study, our experimenter categorized the error that led to incorrect completion. For most tasks, we expected participants to make specific errors. For example, we expected participants doing the Jana task with WW to fail to realize that there was a rule conflict. Most errors we observed were the ones we expected, but we did observe some unexpected errors, like participants finding the wrong Handouts folder in the Piano task or creating a new user named "charles" instead of using the existing "charles" in the Charles task.

7.5.2.1 Charles

Hypotheses

We designed the Charles task to show the advantage our specificity semantics has over the Windows semantics when ALLOW rule exceptions to a group DENY rule are desired. Our hypotheses are:

1. GS will have a higher accuracy rate than GW or WW.
2. GS will have a shorter mean time-to-task-completion than GW or WW.

Results

GS had a higher accuracy rate (61%) than both GW (11%) and WW (0%). Fisher's exact tests showed these differences to be statistically significant ($a_{GS} > a_{GW}, p = 0.004; a_{GS} > a_{WW}, p < 0.001$). GS had a shorter mean time-to-task-completion (M=184, σ=87) than GW (M=320, σ=117). A t-test showed this result not to be statistically significant ($t(1.89) = -2.04, p = 0.19$), but the lack of significance is not surprising given that only 2 GW participants correctly completed the task. No WW participants correctly completed the task, so we cannot compute a mean time-to-task-completion.

Discussion

The superior accuracy rate observed in GS for the Charles task compared to GW and WW illustrates the semantics advantage that comes with an individual being able to have exceptions to the group rules.

The 61% accuracy rate amongst GS participants is notably low, given how simple the task should be for participants in this condition. An analysis of the errors made in the Charles task indicates they are largely due to a combination of misinterpreting the task statement and failing to understand that the background color of subgrid squares indicates effective permissions. Of the seven participants who incorrectly completed the Charles task, five set rules allowing READ access to the entire Lyrics folder to both Charles and Carl. These participants apparently misinterpreted the instruction, "...make sure [Charles] can still read the same files ... that his good friend Carl can read," to mean that Charles and Carl should both be allowed to read all files in the Lyrics folder, rather than the intended meaning that Charles's access should match Carl's existing access. In any case, since Carl has some individual DENY exceptions on files within the Lyrics folder, his file-level exceptions take precedence over the folder-level ALLOW rule, and he and Charles end up with different permissions. Participants could have noticed that Charles and Carl had different effective permissions, but apparently either failed to notice or misunderstood the meaning of the subgrid square background color. The other two incorrect completions were due to an off-by-one-row error and an error adding Charles to the

Alumni group, in which the participant created a new user named “charles” instead of using the existing “charles” as we intended.

7.5.2.2 Kent

Hypotheses

We designed the Kent task to show a drawback our specificity semantics has when ALLOW exceptions to a group DENY rule are not desired. We expect our semantics not to perform as well as the Windows semantics for the Kent task, but we expect the Grid to help participants notice that the ALLOW exceptions are present. On balance, we are not sure how much the Grid presentation will serve to overcome the drawback of the semantics. Our hypotheses are:

1. GS will have a lower accuracy rate than GW. (Since GS has a semantics disadvantage but no presentation advantage over GW.)

2. GS might have a lower accuracy rate than WW.

3. GS will have a longer mean time-to-task-completion than GW or WW.

Results

GS had a lower accuracy rate (50%) than both GW (100%) and WW (94%). Fisher’s exact tests showed these differences to be statistically significant ($a_{GS} < a_{GW}, p = 0.001; a_{GS} < a_{WW}, p = 0.007$). (Note, however, that under the stricter accuracy scoring method, the difference between GS and WW would not be statistically significant.) GS had a longer mean time-to-task-completion (M=155, σ=53) than GW (M=113, σ=31), but a t-test showed this result not to be statistically significant ($t(13.2) = 2.26, p = 0.04$). GS had a shorter mean time-to-task-completion (M=155, σ=53) than WW (M=202, σ=79), but a t-test showed this result not to be statistically significant ($t(19.2) = -1.55, p = 0.14$).

Discussion

As expected, we observed a Windows semantics advantage in the higher accuracy rates for the GW and WW conditions. The Grid’s presentation advantages over the Windows list-of-rules presentation did not overcome the semantics disadvantage. Time-to-task-completion differences tell us little, as they were not statistically significant.

7.5.2.3 Lance

Hypotheses

We designed the Lance task to test the behavior of our specificity semantics in the presence of a two-dimensional conflict when a DENY decision is desired. We expected the GS interface to perform better than the GW and WW interfaces, and expected GW to perform better than WW because of the presentation advantages of the Grid. Our hypotheses are:

1. GS will have a higher accuracy rate than GW or WW.

2. GW will have a higher accuracy rate than WW.

3. GS will have a shorter mean time-to-task-completion than GW or WW.

4. GW will have a shorter mean time-to-task-completion than WW.

Results

GS had a higher accuracy rate (100%) than both GW (39%) and WW (0%). Fisher's exact tests showed these differences to be statistically significant ($a_{GS} > a_{GW}, p < 0.001; a_{GS} > a_{WW}, p < 0.001; a_{GW} > a_{WW}, p = 0.007$). GS had a shorter mean time-to-task-completion (M=87, σ=44) than GW (M=118, σ=52), but a t-test showed this result not to be statistically significant ($t(10.6) = -1.35, p = 0.21$). No WW participants correctly completed the task, so we cannot compute a mean time-to-task-completion.

Discussion

The superior accuracy rate observed in GS for the Lance task compared to GW and WW illustrates the semantics advantage that comes when the semantics chooses the desired conflict resolution by default. The 0% accuracy in WW illustrates the presentation disadvantage of a list-of-rules interface compared to the Grid, which, even with the Windows semantics, was able to achieve a 39% accuracy rate. Participants in the WW condition did not notice that Lance's effective permissions were incorrect.

7.5.2.4 Adria

Hypotheses

We designed the Adria task to test the behavior of our specificity semantics in the presence of a two-dimensional conflict when an ALLOW decision is desired. Since the specificity semantics requires extra steps for correct task completion in the Adria task, we expected the GS interface to perform worse than the GW interface. As in the Kent task, we expected the GS interface's presentation advantage might enable it to overcome its semantics disadvantage compared to the WW interface. Our hypotheses are:

1. GS will have a lower accuracy rate than GW. (Since GS has a semantics disadvantage but no presentation advantage over GW.)

2. GS might have a lower accuracy rate than WW.

3. GS will have a longer mean time-to-task-completion than GW or WW.

Results

GS had a lower accuracy rate (78%) than GW (83%), but a higher accuracy rate than WW (56%). Fisher's exact tests showed neither of these differences to be statistically significant ($a_{GS} < a_{GW}, p = 1; a_{GS} \neq a_{WW}, p = 0.29$). GS had an essentially equal mean time-to-task-completion (M=119, σ=61) compared to GW (M=120, σ=84), and a t-test showed the small difference not to be statistically significant ($t(27.0) = 0.09, p = 0.92$). GS had a shorter mean time-to-task-completion than WW (M=202, σ=101), and a t-test showed this result to be statistically significant ($t(21.2) = -2.75, p = 0.01$).

Discussion

Surprisingly, GS did not perform significantly worse than GW or WW for the Adria task, despite a semantics disadvantage. An analysis of errors made in the WW condition showed that the eight participants who incorrectly completed the task added extraneous access rules for Adria. They apparently did not realize that, once Adria had been removed from the Opera Instructors group, her effective permissions were already set correctly. This is perhaps not surprising given how difficult it is to check effective permissions in the Windows interface. Only three participants made the same error in the GW interface, which allowed participants to see that Adria's effective permissions were already correct after her removal from Opera Instructors. The result showing no statistically significant difference in accuracy between GS and GW and GS and WW suggests that, for the Adria task, the GS's presentation advantage combined with the ease of overcoming the two-dimensional conflict with our specificity semantics did allow GS to overcome its semantics disadvantage.

7.5.2.5 Pablo

Hypotheses

We designed the Pablo task to test the behavior of our specificity semantics in the presence of a group conflict. The Pablo task was structured similarly to the Jana task—Pablo is a member of two groups, one of which is allowed access to a file and the other of which is denied access to that file. Detecting the conflict is easier when effective permissions are visible, and resolving the conflict is easier with our specificity semantics than with Windows. So, we expect GS to perform best, followed by GW, followed by WW. Our hypotheses are:

1. GS will have a higher accuracy rate than GW or WW.
2. GW will have a higher accuracy rate than WW.
3. GS will have a shorter mean time-to-task-completion than GW or WW.
4. GW will have a shorter mean time-to-task-completion than WW.

Results

GS had a higher accuracy rate (94%) than GW (22%) and WW (39%), and Fisher's exact tests showed the differences between conditions to be statistically significant ($a_{GS} > a_{GW}, p < 0.001; a_{GS} > a_{WW}, p < 0.001$). (Note, however, that under the stricter accuracy scoring method, the difference between GS and WW would not be statistically significant.) GW had a lower accuracy rate than WW, but the difference was not statistically significant. GS had a shorter mean time-to-task-completion (M=115, σ=63) compared to GW (M=240, σ=154) and WW (M=185, σ=97), but t-tests showed these differences not to be statistically significant ($ttc_{GS} < ttc_{GW}, t(3.9) = -1.88, p = 0.14; ttc_{GS} < ttc_{WW}, t(9.8) = -1.69, p = 0.12; ttc_{GW} < ttc_{WW}, t(5.8) = 0.57, p = 0.59$).

Discussion

As expected, GS performed best in accuracy and in mean time-to-task-completion (although the time-to-task-completion differences were not statistically significant). Surprisingly, GW did not outperform WW. An analysis of the errors participants made with the GW interface indicates participants would commonly set an access rule stating that Pablo was allowed to read the assignment4.pdf file, but were confused by the difference between the access rule (shown as a light green dog-ear in the subgrid square corresponding to Pablo, READ, and assignment4.pdf) and the effective permission (shown as a red background in the subgrid square). Of course, under specificity semantics, the specific rule applying to Pablo would take precedence over the existing group rules, so it would not be important for participants to understand the distinction between access rule and effective permission, since both would be the same (i.e., both the subgrid dog-ear and background would be green).

7.5.2.6 Piano

Hypotheses

We designed the Piano task to test the group creation functionality of the interfaces and the behavior of the semantics in the presence of a group conflict. Our expectations for the Piano task were similar to those for the Pablo task, except that we expected lower accuracy rates over all three conditions due to errors in the group creation process. Our hypotheses are:

1. GS will have a higher accuracy rate than GW or WW.
2. GW will have a higher accuracy rate than WW.
3. GS will have a shorter mean time-to-task-completion than GW or WW.
4. GW will have a shorter mean time-to-task-completion than WW.

Results

GS had a higher accuracy rate (44%) than GW (33%) and WW (11%), but Fisher's exact tests showed none of the differences in accuracy rates amongst the conditions to be statistically significant ($a_{GS} > a_{GW}, p = 0.73; a_{GS} > a_{WW}, p = 0.06; a_{GW} > a_{WW}, p = 0.23$). GS had a shorter mean time-to-task-completion (M=175, σ=58) compared to GW (M=217, σ=73) and WW (M=336, σ=209), but t-tests showed differences amongst the interfaces not to be statistically significant ($ttc_{GS} < ttc_{GW}, t(10.0) = -1.05, p = 0.32; ttc_{GS} < ttc_{WW}, t(1.1) = -1.21, p = 0.42; ttc_{GW} < ttc_{WW}, t(1.2) = -0.77, p = 0.56$).

Discussion

As expected, GS performed best in accuracy and in mean time-to-task-completion, although the results were not statistically significant. Accuracy rates overall were quite low, below 50% in all conditions. We attribute the low accuracy rates to the difficulty of the task. An error analysis showed that while most participants successfully formed the new Piano Students 2008 group and created an access rule granting the group access to the Piano\Handouts folder, eight GS participants, nine GW participants, and 16 WW participants failed to notice that a group conflict existed, so never took steps to resolve it.

7.5.2.7 Troublemakers

Hypotheses

The troublemakers task was designed to show the Grid's support for visual searches for anomalies in a large section of the effective policy—a search for a "needle in a haystack." We expect the Grid's whole effective policy display to enable such searches, and expected that the Windows display of one resource and principal at a time to make such a search virtually impossible. Our hypotheses are:

1. GS and GW will have higher accuracy rates than WW. (Semantics did not come into play in the Troublemakers task, so we did not expect a different between GS and GW.)

2. WW will have a near-zero accuracy rate. (Hence, meaningful time-to-task-completion comparisons will not be possible.)

Results

The accuracy rates for GS (56%) and GW (39%) were higher than the accuracy rate for WW (0%). Fisher's exact tests showed the differences in accuracy rates between GS and WW (p<0.001) and GW and WW (p=0.007) to be statistically significant. The WW accuracy rate was, as expected, zero.

Discussion

As expected, the Grid interfaces enabled participants to find an anomaly in the effective policy and fix it, and the Windows interface did not. Accuracy rates were still fairly low; an error analysis revealed that GS and GW participants who did not correctly complete the task failed to realize they needed to expand the Troublemakers group and the Opera\Admin folder to find the permissions anomaly. Most of these participants set a DENY rule for the Troublemakers group on the root Classes folder, but failed to realize that the yellow background color of the subgrid square corresponding to Troublemakers' READ access to the Classes folder meant that there was still an ALLOW exception to the group-level DENY rule.

7.5.2.8 Assignment

Hypotheses

We designed the Assignment task to test the semantics following a file move. Because of the potential surprise changes to effective policy that can arise with the Windows semantics, we expected the specificity semantics to be less confusing to participants and thus lead to better performance. Our hypotheses are:

1. GS will have higher accuracy rates than GW and WW.
2. GS will have shorter mean time-to-task-completion than GW and WW.

Results

The accuracy rate for GS (89%) was higher than the accuracy rates for GW (22%) and WW (33%). Fisher's exact tests showed the differences in accuracy rates between GS and GW (p<0.001) and GS and WW (p=0.002) to be statistically significant. The mean time-to-task-completion for GS (M=103, σ=42) was marginally longer than that for GW (M=98, σ=3), but the difference was not statistically significant ($t(15.8) = -0.18, p = 0.86$). The GS mean time-to-task-completion was shorter than that for WW (M=194, σ=60), and the difference was statistically significant ($t(11.2) = -3.93, p = 0.002$).

Discussion

As expected, the Windows semantics led to a surprise policy change that most participants did not notice. Our semantics does not cause such a surprise, so GS participants completed the task correctly.

7.5.2.9 Syllabus

Hypotheses

We designed the Syllabus task to test the downside of the aspect of our semantics that led to the gains in the Assignment task. In the Syllabus task, the task goal calls for a moved file to retain permissions it had under its prior parent. We expected the Windows semantics to perform best, but expected the Grid's presentation advantages would help the GS interface overcome at least part of its semantics disadvantage. Our hypotheses are:

1. GS might have lower accuracy rates than GW and WW.
2. GS might have longer mean time-to-task-completion than GW and WW.

Results

The accuracy rate for GS (72%) was lower than the accuracy rates for GW (94%) and WW (100%). However, Fisher's exact tests showed the differences in accuracy rates between GS and GW (p<0.17) and GS and WW (p=0.05) not to be statistically significant. The mean time-to-task-completion for GS (M=156, σ=54) was longer than that for GW (M=77, σ=25), and a t-test showed the difference to be statistically significant ($t(26.0) = 4.97, p < 0.001$). The GS mean time-to-task-completion was longer than that for WW (M=101, σ=60), and the difference was statistically significant ($t(28.0) = 3.20, p = 0.003$).

Discussion

With the Windows semantics, the Syllabus task is a simple matter of moving a file—no change to file permissions is necessary. Thus, it is not surprising to see very high accuracy rates for GW and WW. It is also not surprising to see that the mean times-to-task-completion for GW and WW were statistically significantly shorter than those for GS. Interestingly, however, the GS accuracy rates were fairly high. They were not statistically significantly different from the GW or WW accuracy rates (although there was a nearly significant difference with WW). The dropoff in performance due GS's semantics disadvantage in the Syllabus task was not as great as the dropoff in performance due to WW's semantics disadvantage in the Assignment task. While we must be careful about making precise numerical comparisons between the tasks, since their structure was not identical, it does appear that the Grid's presentation advantages help mitigate the downside of its semantics for the Syllabus task.

7.5.2.10 Jana

Hypotheses

Like the Pablo and Piano tasks, the Jana task tested the behavior of the two semantics in the presence of a group conflict. As with those tasks, we expect GS to perform best, followed by GW, followed by WW. Our hypotheses are:

1. GS will have a higher accuracy rate than GW or WW.
2. GW will have a higher accuracy rate than WW.
3. GS will have a shorter mean time-to-task-completion than GW or WW.
4. GW will have a shorter mean time-to-task-completion than WW.

Results

GS had a higher accuracy rate (89%) than GW (17%) and WW (56%). Fisher's exact tests showed the difference between GS and GW accuracy rates to be statistically significant (p<0.001), but showed the difference between GS and WW accuracy rates not to be significant (p=0.06). (Note, however, that under the stricter accuracy scoring method, the difference between GS and WW would be statistically significant.) GW had a lower accuracy rate than WW, and the difference was nearly, but not quite, statistically significant after the correction for multiple testing (p=0.035). GS had a shorter mean time-to-task-completion (M=60, σ=23) compared to GW (M=201, σ=118) and WW (M=253, σ=94). The difference in mean time-to-task-completion between GS and GW was not statistically significant ($t(2.1) = -2.40, p = 0.13$), but the difference between GS and WW was ($t(19.2) = -10.1, p < 0.001$). The difference in between GW and WW was not significant ($t(2.3) = -0.74, p = 0.53$).

Discussion

GS performed best in accuracy and in mean time-to-task-completion. This result was expected, especially in light of the results for the Jana task in the study described in Chapter 6, in which the Grid accuracy rate was 100% compared to 6% for Windows. What is surprising here is that the Windows accuracy rate is as high as it is, at 56%. We attribute the relatively high WW accuracy rate to the additional training we gave participants (specifically the training in how to check effective permissions in WW), the ability to move principals in and out of groups (which they were not allowed to do in the prior study), and the somewhat more advanced participant pool.

As in the Pablo task, GW, somewhat surprisingly, did not outperform WW. Also as in the Pablo task, an analysis of the errors participants made with the GW interface indicates they were confused by the distinction between access rules and effective permissions. In Windows, participants were somewhat less confused by the distinction because effective permissions are in a completely different view from access rules.

7.5.2.11 Clayton and Quincy

We did not perform any specific statistical testing for the Clayton and Quincy tasks. The tasks were borrowed from our prior study on the file-permissions Expandable Grid interface. We included them in this study mainly to provide a wider variety of tasks over which to compute experiment-wide results. We did not expect much difference between the interfaces for these tasks, and the tasks were not related to testing the semantics or the new functionality of the Grid.

7.5.2.12 Experiment-wide discussion

Our results show that the policy semantics underlying a file-permissions interface has a significant effect on task-completion accuracy. The precise effect of the semantics depends on what the task goals are. If task goals are aligned with a semantics, such that the semantics yields the desired effective policy by default, accuracy rates are likely to be higher than if the semantics requires an author to take additional action to change the decision. In the Charles, Kent, Lance, Assignment, and Syllabus tasks, we saw that the semantics that yielded the correct effective policy by default led to the highest accuracy rates.

No semantics can always yield the desired effective permissions, as we designed our inverse task pairs (Charles/Kent, Lance/Adria, and Assignment/Syllabus) to show. However, we argue that our specificity semantics is superior to the Windows semantics because, even in a situation where specificity semantics is not yielding the desired effective permission, it is easy to change the effective permission with a specific rule. Contrast this to the Windows deny-precedence semantics, which forces an author in some conflict situations to remove a user from a group or change rules applying to the whole group. The Adria task shows a situation in which, although our specificity semantics does not yield the desired decision by default, getting the task right is a simple matter of adding a rule more specific than those in conflict. The Pablo, Piano, and Jana tasks further illustrate the advantage that a specificity semantics has over a deny-precedence semantics in overcoming rule conflicts.

The Assignment and Syllabus tasks exercised a specific difference between our semantics and the Windows semantics, namely, the timing of when a moved resource inherits its new parent's access rules. The Assignment task illustrates that, as we expected, the Windows semantics can cause surprises that lead to errors. On the other hand, in the Syllabus task, GS accuracy rates remained fairly high even though the semantics did not yield the desired effective permission by default. Since our semantics does not cause the surprises that the Windows semantics does, but can still support high accuracy rates even when it does not yield the desired effective permission by default, we conclude that allowing a moved resource to inherit its new parent's access rules immediately upon being moved is preferable, from a usability standpoint, to delaying the rule inheritance until the next access-control policy change.

As we observed in the prior file-permissions Expandable Grid study described in Chapter 6, the Grid is a better way to display a policy to authors than is a list of rules, at least in the scenarios that we tested. In the Policy Semantics Study #2, the Grid's presentation advantages helped GS overcome semantics disadvantages in the Adria and Syllabus tasks. We also observed in the Lance task that although GW and WW were both at the same semantics disadvantage, GW had a higher accuracy rate, which we attribute to the Grid's presentation advantages. The Troublemakers task, which requires a visual search through a large portion of a policy, is an example of a task that is virtually impossible to complete correctly in Windows but is easily completed using the Grid.

We did observe some usability problems with the Grid, some familiar from our prior study, some new. Familiar problems included finding the wrong object when searching (finding a similarly named object, e.g., the Music 101\Handouts folder instead of the Piano\Handouts folder) and mouse slippage (in which a participant using the Grid would find a column they wanted, then while moving the mouse down the column to find a row they wanted, the mouse would slip into an adjacent column without the participant noticing).

One new usability problem had to do with the new "dog-ear" feature. Participants often did not understand the distinction between access rules and effective permissions. They would frequently interpret dog-ears as representations of the effective permissions and would ignore background colors. We anticipated this problem, and tried to mitigate it with a carefully designed legend. We referred to access rules as "default rules" to suggest that they were inherited by default but could be overridden. In the legend, we depicted every possible combination of dog-ear color on background color. Many participants did see and use the legend, but not all understood it. We did not have this problem in our prior Grid study because that Grid did not display access rules at all. While it would be nice not to display access rules at all, we believe that when dynamic changes are allowed, it is necessary to represent access rules in some way. We may need more work to determine the best way to convey the distinction between access rules and effective permissions to policy authors.

Another new problem we observed was participants not understanding that, when seeing a yellow square, it was necessary to expand one or both of the principal and resource trees to see specifically what permissions underlay the yellow. Even though we had a footnote in the legend explaining that one or both trees should be expanded to get detail on yellow squares, some participants did not notice the footnote in the legend. It is somewhat surprising that we did not observe this problem in our prior study. Training in both the prior study and this study explained the meaning of yellow squares, but the message about yellow squares may have been lost to some participants in this study because we presented much additional material in the training for this study.

Our study design had some limitations that are worth considering. Our results tell us that the success of any semantics depends largely on the requirements of the particular tasks that an author performs. Since we do not have data on the frequency with which particular tasks are performed, we cannot say conclusively that our specificity semantics is superior to the Windows semantics. We can only argue that its behavior more closely follows the four desired semantics properties and makes it easier to fix situations in which its default behavior is not what is desired compared to the Windows semantics.

Our study created scenarios that participants could complete within a few minutes in our lab. We were not able to recreate certain real-life scenarios in which policy-authoring may happen periodically over long periods of time. Particularly difficult to simulate in the lab are scenarios in which an author must remember the intention behind a setting that may have been made weeks or months before.

7.6 Conclusion

We have demonstrated significant usability improvements from a file-system access-control semantics centered on specificity precedence compared to a semantics centered on deny precedence. Our specificity semantics adheres to the properties of direct manipulation, exception-rule preservation, order independence, and fail safety more closely than the Windows NTFS access-control semantics does. We have implemented our new semantics in a simulated Windows file system and run the file-permissions Expandable Grid on top of it. We have shown in a user study that the semantics provides substantial usability gains for tasks involving rule conflicts, and that it may make certain operations, like file moves, less prone to access-control policy errors.

No semantics can yield the right effective policy in all situations, but a specificity semantics should work well in general because a specific rule would be unlikely to have been set unless it was meant to take effect. Moreover, a specificity semantics makes it easier to overcome the situations in which the semantics does not yield the desired effective policy by default.

We have argued that our specificity semantics is better than the Windows semantics for three reasons: that specificity provides a better way to resolve a rule conflict than removing a user from a group or changing a group's access rules; that defaulting to a DENY decision is safer than arbitrarily favoring the resource dimension over the principal dimension in resolving two-dimensional conflicts; and that immediately allowing a moved resource to inherit its new parent folder's access rules is more predictable than deferring inheritance until a change is made to the new parent's or moved resource's access control list.

Our results suggest that, whatever the underlying semantics, the Grid is a superior way to help policy authors understand how the semantics works and complete tasks correctly than is the Windows list-of-rules interface.

CHAPTER 8

Expandable Grid Interface for Displaying P3P Policies

This chapter is largely a reproduction of a paper co-authored with Patrick Kelley, Aleecia McDonald, and Lorrie Cranor and submitted to the 2008 Workshop on Privacy in the Electronic Society.

Presenting website privacy policies to consumers in a clear and concise manner is important from at least two perspectives. First, from the perspective of consumer protection, fair information practice principles require that consumers be informed of how websites will use consumers' personal information. The United States Federal Trade Commission (FTC) names notice/awareness, i.e., giving consumers notice of an entity's information practices, as the first of its five core principles of privacy protection [45]. Second, from the perspective of websites trying to gain consumers' trust, providing clear notice of privacy practices may help allay consumer concerns about misuse of their personal data [27]. Moreover, for e-commerce websites, consumers may be willing to pay a premium if presented with a prominent display of consumer-friendly privacy practices [131].

However, despite the importance of clear presentation, privacy policies are usually presented in legalistic, convoluted language [100], and are often written at a college or higher reading level [7]. Researchers, industry groups, and privacy advocates have proposed other methods for presenting privacy policies, but there is no consensus on an effective presentation format. To make matters worse, presentations vary from website to website; thus, from the consumer's point of view, every privacy policy is as hard to read as the last one, and it is very difficult to compare privacy policies across competing websites.

The Platform for Privacy Preferences (P3P) was designed to address some of the drawbacks to natural language privacy policies [39, 40]. P3P is an eXtensible Markup Language (XML) based machine-readable language for expressing website privacy policies. It enables websites to specify policies in a uniform manner that can be read and presented by user agents, such as a Web browser or a policy-display application like Privacy Bird [41]. Many websites provide P3P policies; a 2007 study found that 28% of the top 75 dot-com domains had been P3P-enabled [38]. The P3P specification does not specify a presentation format.

We introduce the P3P Expandable Grid, an interactive format for presenting P3P policies to website visitors. The P3P Expandable Grid is based on the Expandable Grid concept, presented in Chapter 5. In Chapter 6, we described our application of the Expandable Grid concept to a user interface for displaying and authoring file permissions policies and showed that our Expandable-Grid-based interface was very effective for a wide range of file-permissions policy-authoring tasks. Our result suggests that the Expandable Grid idea might work similarly well for displaying P3P policies. We designed and implemented the P3P Expandable Grid to see if the Expandable Grid would fulfill its promise as a P3P policy presentation format.

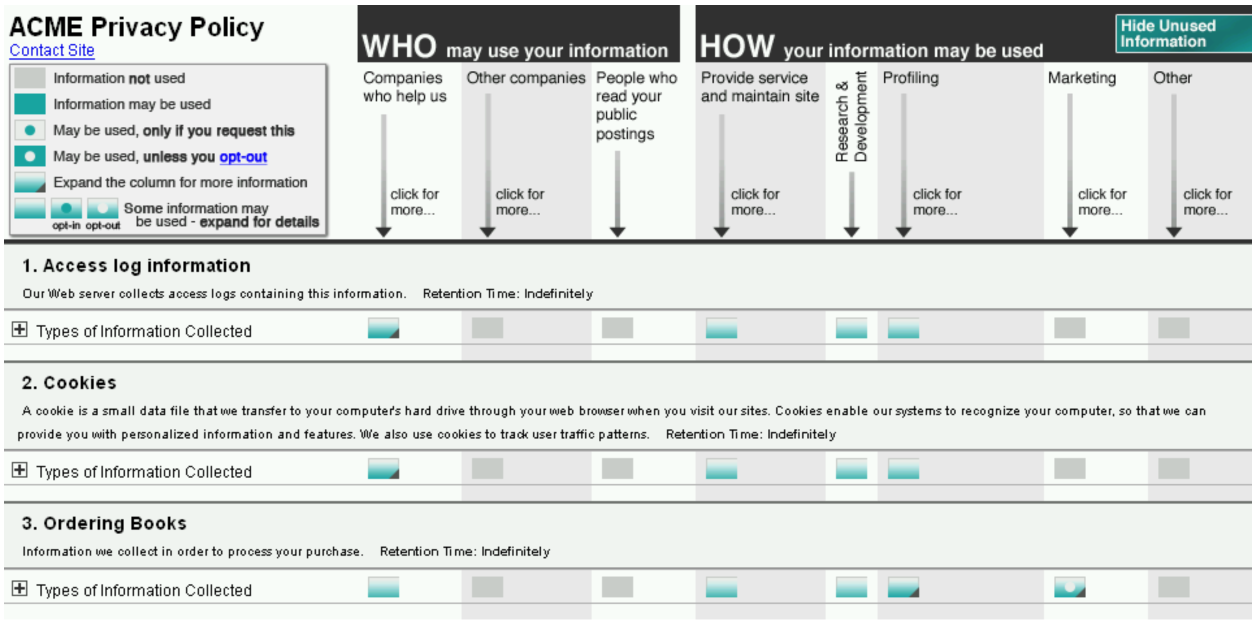
We evaluated the P3P Expandable Grid in two studies, one Web-based and one conducted in our lab. In the Web-based study, 520 participants viewed website privacy policies either in natural language or in the P3P Expandable Grid and were asked comprehension questions about the policies. Participant performance was poor in both conditions, and, to our surprise, was generally worse with the P3P Expandable Grid. To explain this unexpected result, we conducted a lab study in which we collected detailed video and think-aloud audio data from 12 participants who viewed the same policies in both the natural language and P3P Expandable Grid formats and answered the same comprehension questions. We identified specific usability problems with the P3P Expandable Grid as a tool for displaying privacy policies to consumers. These usability problems, such as lack of a focal point, unintuitive organization of policy elements, and confusing icons and terminology, suggest both specific improvements that might make the P3P Expandable Grid an effective tool and general lessons as to when and how to apply the Expandable Grid concept to other policy domains. We discuss these suggested improvements and general lessons. We also note that while the current implementation of the P3P Expandable Grid did not work well for conveying privacy policies to consumers who had never seen the Grid before, it might be of use as an authoring tool for P3P experts or as the basis for a standardized privacy policy presentation.

8.1 System description

Our system for displaying P3P policies, the P3P Expandable Grid, takes the elements of a P3P policy and maps them onto the conceptual framework of the Expandable Grid visualization. Here we describe the relevant elements of P3P, the Expandable Grid concept, and our implementation of the P3P Expandable Grid.

8.1.1 P3P

P3P defines a set of data practices in which an organization with a website could potentially engage. A specific organization's P3P policy states which of these practices the organization actually engages in. For example, one potential data practice is collecting a consumer's online contact information (e.g., email address) and sharing it with other companies to contact the consumer for marketing purposes. Another potential data practice is collecting a consumer's web navigation patterns and using them to improve a website's layout. Each potential data practice defined by P3P consists of three primary data-specific assertions: data category, recipient, and purpose. (Actually, P3P defines six types of data-specific assertions, but to simplify this discussion, we leave out the infrequently-used non-identifiable assertion, the almost-invariant retention assertion (i.e., websites with P3P policies almost invariably retain data "indefinitely"), and the free-text human-readable consequence assertion.) So, in the first example data practice above, "online contact information" is the data category, "other companies" is the recipient, and "contact for marketing" is the purpose. P3P defines 17 data categories, 12 purposes, and six recipients. It also defines a hierarchy of 31 data elements, each of which can have numerous sub-elements (data elements are items within the data categories, such as email address or home phone number). Any combination of data element (or category), purpose, and


ACME Privacy Policy
Contact Site
Information not used
Information may be used
May be used, only if you request this
May be used, unless you opt-out
Expand the column for more information
Some information may be used - expand for details
opt-in
opt-out
WHO may use your information
Companies who help us
Other companies
People who read your public postings
HOW your information may be used
Hide Unused Information
Provide service and maintain site
Research & Development
Profiling
Marketing
Other
click for more...

1. Access log information

Our Web server collects access logs containing this information. Retention Time: Indefinitely

⊞ Types of Information Collected

2. Cookies

A cookie is a small data file that we transfer to your computer's hard drive through your web browser when you visit our sites. Cookies enable our systems to recognize your computer, so that we can provide you with personalized information and features. We also use cookies to track user traffic patterns. Retention Time: Indefinitely

⊞ Types of Information Collected

3. Ordering Books

Information we collect in order to process your purchase. Retention Time: Indefinitely

⊞ Types of Information Collected

Figure 8.1: *Screenshot of the P3P Expandable Grid in unexpanded form. The column groups labeled "WHO" and "HOW" show P3P recipients and purposes, respectively. The rows show three P3P statements with their header boxes and the roots of their data hierarchies (the roots are labeled "Types of information collected"). The column headers and data hierarchies can be expanded and contracted to show more or fewer P3P elements.*

recipient constitutes a potential data practice. A P3P policy consists of one or more statements, which are XML elements that each contains a group of data elements that are all to be handled similarly. A statement thus declares a set of the potential data practices an organization actually engages in.

There are four distinct P3P policy outcomes that we name REQUIRED, NOT-USED, OPT-IN, and OPT-OUT. A P3P statement maps each of the potential data practices to a policy outcome. REQUIRED outcomes indicate data practices in which an organization engages; NOT-USED outcomes indicate data practices in which an organization does not engage; OPT-IN outcomes indicate data practices in which an organization engages if a consumer requests to be subjected in the practice; and OPT-OUT outcomes indicate data practices in which an organization engages unless the consumer requests not to be subjected to the practice.

In the preceding description, we have simplified some of the details of P3P. A full description of the language can be found in the book *Web Privacy with P3P* [40] or in the P3P standard [39].

8.1.2 P3P Expandable Grid design

Since P3P data practices are defined by the three primary data-specific assertions, we used hierarchical structures of the P3P data categories, purposes, and recipients as the labels along the axes of our P3P Expandable Grid. Our data hierarchy uses the data categories as high-level nodes, and places the P3P data element hierarchy under the categories as per the P3P specification [39]. We put an expandable tree representation of the data hierarchy along the vertical (left-hand) axis of our grid. P3P defines its purposes and recipients in flat lists of categories, but we added a layer of structure to simplify presentation. For example, we grouped the P3P “contact” and “telemarketing” purposes into one higher-level “marketing” category. Since, after putting data categories along the vertical axis, we had only one axis left to represent two hierarchies, we put both the recipient and purpose hierarchies next to each other on the horizontal (top) axis. The layout of the P3P Expandable Grid can be seen in Figures 8.1 and 8.2.

While putting recipient and purpose on the same axis would seem to restrict our ability to represent certain policies, namely those in which both recipient and purpose vary for the same data element, P3P itself has the same restriction, and requires multiple statements to represent such policies. Since our grid design can represent multiple statements, we are able to get around this restriction in the same way the underlying P3P language does. So, for example, a policy which allows email addresses to be collected by the company issuing the policy for the purpose of site administration, but also allows email addresses to be collected and shared with other companies for the purpose of marketing, will require multiple P3P statements.

The grid itself consists of squares at the intersections of each row representing a data element and each column representing a recipient or a purpose. The squares are colored according to the P3P policy outcome for each data practice defined by the combinations of data elements, recipients, and purposes. For each data element, the grid squares corresponding to recipients who use the data element or to purposes for which the element is used are colored teal; grid squares corresponding to recipients that do not use the data element or purposes for which the data element is not used are colored grey. We chose teal as a neutral color with no obvious conventional meaning; we were concerned that green or red might imply “good” or “bad” judgments about policy content. Grid squares at the intersection of non-leaf nodes of the hierarchies potentially represent multiple distinct outcomes of the leaf nodes beneath them, so they are colored with a teal-white gradient if there are both teal and grey squares beneath them, just as yellow squares in the file permissions Expandable Grid indicate a mixture of allow and deny access in the nodes beneath them. The different colored squares are shown in Figure 8.2.

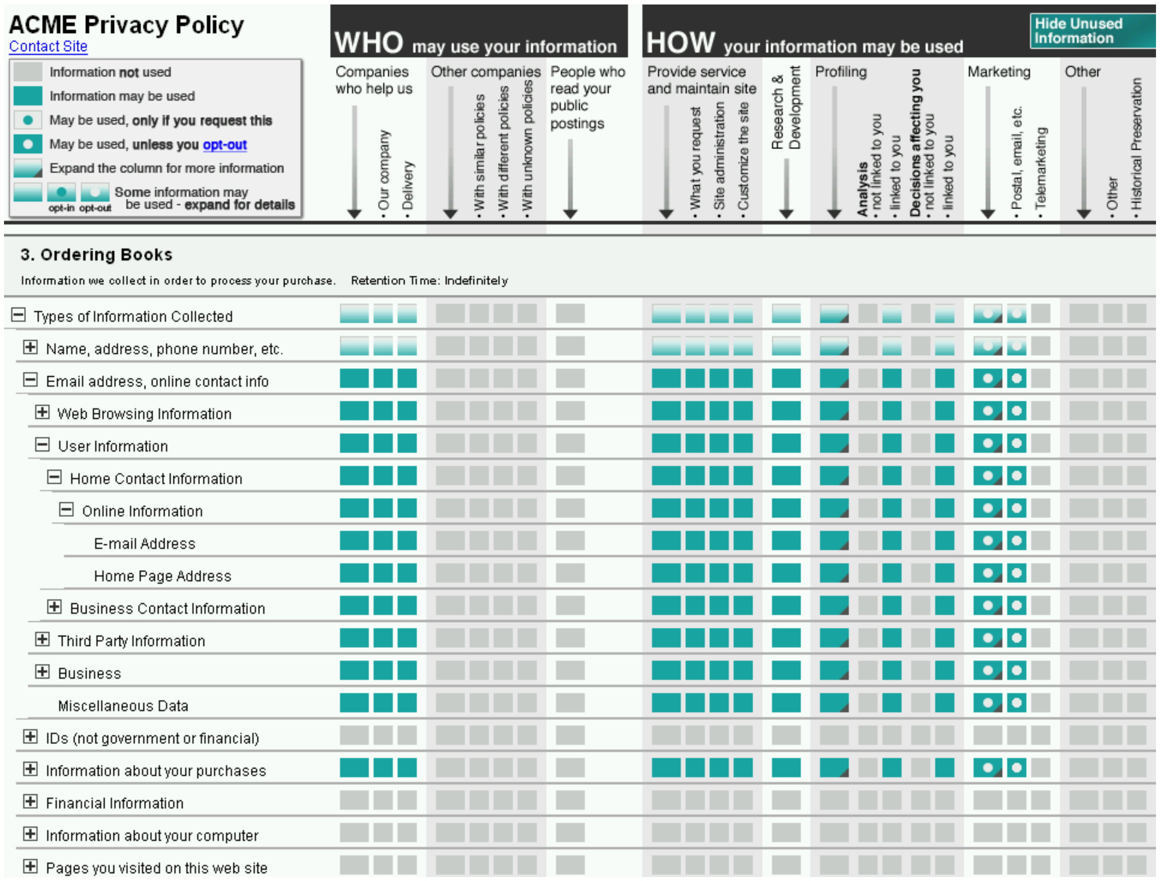
Teal and grey squares cover the REQUIRED and NOT-USED outcomes of P3P policies, but not the OPT-IN and OPT-OUT outcomes, so we added a dot notation to indicate data practices for which the consumer was given the choice to opt into or out of. Squares are colored with a teal dot on a white background to indicate an OPT-IN outcome and a white dot on a teal background to indicate an OPT-OUT outcome. Dots over gradients indicate squares corresponding to non-leaf nodes under which there are some OPT-IN or OPT-OUT outcomes. Squares representing the OPT-OUT outcome can be seen in Figure 8.2.

All square designs may also contain a small black dog-ear symbol in the lower right-hand corner, indicating the cell may be clicked to expand its column. The dog-ear symbol can be seen on some of the squares in Figure 8.2.

Metadata associated with each P3P statement is contained in a header box of text above the data hierarchy. Multiple statements with header boxes and unexpanded data hierarchies can be seen in Figure 8.1.

Each data hierarchy is initially displayed in its fully collapsed form, i.e., only the root node of the data hierarchy is shown, and a user has to click to expand the root node in order to see data categories and elements (see Figure 8.1). The recipient and purpose hierarchies are also initially shown in their fully collapsed states.

Metadata associated with the entire P3P policy, such as information on opt-out mechanisms and organizational contact information, is in two places: a legend in the upper-left corner of the grid display contains an opt-out link (visible in Figures 8.1 and 8.2), and text below all of the data statements shows organizational contact information and any additional policy metadata (not shown in the figures).



ACME Privacy Policy
Contact Site

⬜ Information **not** used
🟩 Information may be used
⊙ May be used, **only if you request this**
◉ May be used, **unless you** opt-out
▨ Expand the column for more information
▨ ⊙ ◉ opt-in opt-out Some information may be used - **expand for details**

WHO may use your information

HOW your information may be used

Hide Unused Information

3. Ordering Books
Information we collect in order to process your purchase. Retention Time: Indefinitely

	WHO: Companies who help us	• Our company	• Delivery	Other companies	• With similar policies	• With different policies	• With unknown policies	People who read your public postings	HOW: Provide service and maintain site	• What you request	• Site administration	• Customize the site	Research & Development	Profiling	**Analysis** • not linked to you	• linked to you	**Decisions affecting you** • not linked to you	• linked to you	Marketing	• Postal, email, etc.	• Telemarketing	Other	• Other	• Historical Preservation
⊟ Types of Information Collected	▨	▨	▨	⬜	⬜	⬜	⬜	⬜	▨	▨	▨	▨	▨	▨	⬜	▨	⬜	▨	▨	▨	⬜	⬜	⬜	⬜
⊞ Name, address, phone number, etc.	▨	▨	▨	⬜	⬜	⬜	⬜	⬜	▨	▨	▨	▨	▨	▨	⬜	▨	⬜	▨	▨	▨	⬜	⬜	⬜	⬜
⊟ Email address, online contact info	🟩	🟩	🟩	⬜	⬜	⬜	⬜	⬜	🟩	🟩	🟩	🟩	🟩	🟩	⬜	🟩	⬜	🟩	◉	◉	⬜	⬜	⬜	⬜
⊞ Web Browsing Information	🟩	🟩	🟩	⬜	⬜	⬜	⬜	⬜	🟩	🟩	🟩	🟩	🟩	🟩	⬜	🟩	⬜	🟩	◉	◉	⬜	⬜	⬜	⬜
⊟ User Information	🟩	🟩	🟩	⬜	⬜	⬜	⬜	⬜	🟩	🟩	🟩	🟩	🟩	🟩	⬜	🟩	⬜	🟩	◉	◉	⬜	⬜	⬜	⬜
⊟ Home Contact Information	🟩	🟩	🟩	⬜	⬜	⬜	⬜	⬜	🟩	🟩	🟩	🟩	🟩	🟩	⬜	🟩	⬜	🟩	◉	◉	⬜	⬜	⬜	⬜
⊟ Online Information	🟩	🟩	🟩	⬜	⬜	⬜	⬜	⬜	🟩	🟩	🟩	🟩	🟩	🟩	⬜	🟩	⬜	🟩	◉	◉	⬜	⬜	⬜	⬜
E-mail Address	🟩	🟩	🟩	⬜	⬜	⬜	⬜	⬜	🟩	🟩	🟩	🟩	🟩	🟩	⬜	🟩	⬜	🟩	◉	◉	⬜	⬜	⬜	⬜
Home Page Address	🟩	🟩	🟩	⬜	⬜	⬜	⬜	⬜	🟩	🟩	🟩	🟩	🟩	🟩	⬜	🟩	⬜	🟩	◉	◉	⬜	⬜	⬜	⬜
⊞ Business Contact Information	🟩	🟩	🟩	⬜	⬜	⬜	⬜	⬜	🟩	🟩	🟩	🟩	🟩	🟩	⬜	🟩	⬜	🟩	◉	◉	⬜	⬜	⬜	⬜
⊞ Third Party Information	🟩	🟩	🟩	⬜	⬜	⬜	⬜	⬜	🟩	🟩	🟩	🟩	🟩	🟩	⬜	🟩	⬜	🟩	◉	◉	⬜	⬜	⬜	⬜
⊞ Business	🟩	🟩	🟩	⬜	⬜	⬜	⬜	⬜	🟩	🟩	🟩	🟩	🟩	🟩	⬜	🟩	⬜	🟩	◉	◉	⬜	⬜	⬜	⬜
Miscellaneous Data	🟩	🟩	🟩	⬜	⬜	⬜	⬜	⬜	🟩	🟩	🟩	🟩	🟩	🟩	⬜	🟩	⬜	🟩	◉	◉	⬜	⬜	⬜	⬜
⊞ IDs (not government or financial)	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜
⊞ Information about your purchases	🟩	🟩	🟩	⬜	⬜	⬜	⬜	⬜	🟩	🟩	🟩	🟩	🟩	🟩	⬜	🟩	⬜	🟩	◉	◉	⬜	⬜	⬜	⬜
⊞ Financial Information	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜
⊞ Information about your computer	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜
⊞ Pages you visited on this web site	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜

Figure 8.2: *Screenshot of the P3P Expandable Grid in expanded form. In this screenshot, the Grid has been scrolled to show just the third P3P statement in the policy. All column headers and some data hierarchy headers have been expanded. Each square in the grid represents a potential P3P data practice. Grey (light grey in greyscale print) squares indicate data practices in which the organization issuing the policy does not engage; teal (dark grey in greyscale print) squares indicate data practices in which the organization does engage; gradient-colored squares indicate non-leaf nodes, which, if their corresponding rows or columns are expanded, will have both grey and teal squares beneath them; and teal squares with white dots indicate data practices that consumers may opt out of.*

8.2 Related work

Prior work has shown that natural language policies are difficult to read, with an average Flesch Grade Level of 14 (college level), even though only 27% of the US population has a college education [65]. A review of 60 financial privacy policies found they were all "difficult" to read or worse on the Flesch Reading Ease scale [58]. In addition to being difficult to understand, companies use textual ambiguity to obscure data collection practices, for example, they might state they perform a given practice "from time to time" to minimize its perceived importance [100].

The law firm Hunton & Williams took a leadership role in advocating layered online privacy policies as a better format [79]. Layered policies present a one-page summary with a link to more detailed information [35].

The Kleimann Communication Group studied Gramm-Leach-Bliley Act financial statements and developed improved designs for printed policies [105]. After extensive study of multiple presentations, they found a "...table design worked far better in helping consumers easily access, understand, and compare sharing practices" [105].

The Privacy Bird project uses P3P to create a standardized privacy report. The Privacy Bird report uses bulleted lists to summarize information and has a hide/expand feature so Internet users can focus on a high-level summary or drill down for full information about an online policy. Prior research has shown users like the Privacy Bird format and are able to answer comprehension questions about privacy policies using the format, but sometimes make errors because they fail to notice when information is collected on an opt-in or opt-out basis [41].

With the Expandable Grid, we used a table format as suggested by the Kleimann report. The Grid supports the ability to condense information by contracting the columns and rows, which is similar to the design goals of layered policies and Privacy Bird. Grid icons clearly show when information is opt-out or opt-in.

8.3 Method

We conducted two user studies to compare the P3P Expandable Grid format for presenting privacy policies to the natural language format. Participants in our studies answered policy comprehension questions using either a privacy policy written in natural language or the same privacy policy written in P3P and presented with the P3P Expandable Grid.

8.3.1 Web-based user study

The first of our user studies was conducted over the Web. We used a between-participants design with two factors: presentation format and policy length. Format had two levels: Grid and natural language, and length had three levels: short, medium, and long, so there were a total of six conditions. (This study was part of a larger study that included two other formats, layered natural language and Privacy Finder, in the comparison, but we exclude those formats from our discussion here; those results are part of another researcher's work.) We assigned participants randomly to conditions.

8.3.1.1 Participants

We posted advertisements to a variety of online forums to recruit participants to complete our study. Participants received entry into a drawing for a $250 Amazon.com gift certificate. Online advertising forums included the craigslist classified ad site, sweepstakes websites, mailing lists, and personal networks. We also purchased Google adwords, although fewer than ten participants came through Google ads. The variety of advertising venues we used was intended to attract our desired demographic of the general class of Web users. We recruited 786 participants to start the study, of which 520 completed the study and 266 dropped out (see Section 8.4.3 for an analysis of participant dropout rates).

8.3.1.2 Policies used

We presented the same privacy policy to participants in both the Grid and natural language conditions. The policy we chose was a real privacy policy from a major publisher. We chose this policy for the following reasons:

- It was real, and thus representative of a privacy policy a consumer might encounter in practice;
- It was published in both natural language and P3P versions;
- The P3P version had multiple data statements, and we wanted to be able to test the P3P Expandable Grid's ability to show a policy with multiple statements.

We changed references to the publisher's company name in the policy to "Acme" to avoid any associations with the real company from which the policy was taken. Since we were varying policy lengths, we wanted to present long, medium, and short versions of the policy. We used the whole real P3P policy, consisting of eight P3P data statements, as the long version, and we eliminated some of the statements from the whole policy to create a two-statement medium version and a one-statement short version. We also created medium and short versions of the natural language policy that corresponded to the medium and short versions of the P3P policy. Since the real natural language policy organized its content into paragraphs corresponding one-for-one to the P3P policy's data statements, paring down the natural language policy to match the medium and short P3P policies was simply a matter of eliminating the paragraphs that corresponded to the eliminated data statements.

8.3.1.3 Questions used

We asked seven multiple-choice comprehension questions about the privacy policies:

1. *Telemarketing:* Will Acme collect your home phone number and use it for telemarketing?
2. *Cookies:* Does the Acme website use cookies?
3. *Marketing Email:* Does this privacy policy allow Acme to put you on an email marketing list?
4. *Opt-out:* How can you remove yourself from Acme's email list?
5. *Share:* Does this privacy policy allow Acme to share your email address with a marketing company that might put you on their email marketing list?
6. *SSN:* Does the Acme website collect your Social Security number?
7. *Encryption:* If you send your credit card number to Acme do they keep it encrypted to prevent data theft?

For all questions except the Opt-out question, the multiple choice answers were "Yes," "No," or "The policy does not say." For the Opt-out question, multiple choice answers were "Call Acme's customer service," "Send email to Acme," "Click a link to opt out," "You cannot make any choices," and "The policy does not say," and participants were asked to check as many answers as applied.

The Telemarketing question was always presented first and served as a question to allow participants to gain some familiarity with the presentation format they were using. We excluded the Telemarketing question from data analysis.

We took the next four questions, Cookies, Marketing Email, Opt-out, and Share, from Cranor et al. [41], who found that they reflect common consumer privacy concerns.

We designed the last two questions, SSN and Encryption, specifically to test aspects of the Grid. The SSN question tested whether participants could determine that the company does not engage in a particular practice. In the policy we selected, Social Security number is not collected. One advantage of the Grid over natural language is that it specifically shows data practices in which an organization does not engage through the use of grey boxes. Natural language policies generally do not discuss practices in which the organization does not engage, so it is often difficult to determine decisively from a natural language policy that an organization does not engage in a practice. The Encryption question asked for information that actually was not covered in the privacy policy; we expected participants using both formats to struggle to answer the question, but expected it might be easier to see in the Grid layout that encryption is a security feature, not a data practice defined by P3P.

8.3.1.4 Subjective satisfaction questions

After participants completed comprehension questions, we asked a series of questions regarding their subjective satisfaction with the privacy policy presentation. We asked participants to rate their agreement on a scale from 1 (Strongly Agree) to 7 (Strongly Disagree) with the following statements:

1. I feel secure about sharing my personal information with Acme after viewing their privacy practices.
2. I believe Acme will protect my personal information more than other companies.
3. Finding information in Acme's privacy policy was a pleasurable experience.
4. I feel that Acme's privacy practices are explained thoroughly in the privacy policy I read.
5. I feel confident in my understanding of what I read of Acme's privacy policy.
6. This privacy policy was easier to understand than most policies.
7. It was hard to find information in Acme's policy.
8. If all privacy policies looked just like this I would be more likely to read them.

8.3.1.5 Procedure

When participants visited our study website, they were randomly assigned to one of the two formats and one of the three policy lengths. The two formats and three policy lengths made for six total conditions, so we implemented random assignment by assigning each subsequent participant to the next condition in a repeating sequence of the six conditions. After being assigned to a condition, participants were presented a Web page with a comprehension question and a multiple-choice form for completing the question in a frame at the top of the page, and a privacy policy presentation in the assigned format and length in a frame beneath the comprehension question. All participants were presented the Telemarketing question first. After participants answered the Telemarketing question, they were presented the remaining six questions in random order to guard against learning and sequencing effects. Following completion of the comprehension questions, participants were asked the questions listed in section 8.3.1.4 about their subjective satisfaction with the format they used.

8.3.1.6 Data collected

For each participant, we recorded their multiple-choice answers to comprehension questions, the time it took them to answer each question, whether or not they finished the comprehension questions, when they expanded rows or columns in the Grid, and their answers to the questions listed in section 8.3.1.4 regarding subjective satisfaction. We scored each multiple-choice answer as correct or incorrect to compute accuracy rates for each question and each condition.

8.3.2 Lab-based user study

We followed up the Web-based study with a similarly designed study in our laboratory that allowed us to collect detailed qualitative data to help explain the quantitative results of the Web-based study. We recruited 12 participants via a local university newsgroup. Participants answered the same seven comprehension questions used in the Web-based study, but policy presentation conditions were somewhat different. Presentation format was a within-participants variable, so each participant answered all seven comprehension questions twice, once using the Grid format and once using the natural language format. Participants did not view the same policy in both formats, however; if they viewed the short policy in the Grid, they viewed the long policy in natural language, and vice versa. We dropped the medium policy length and used only the short and long policy lengths. We alternated the order in which the two formats were presented, so that six of the participants viewed a policy in the Grid first, and six viewed a policy in natural language first. We asked participants to think aloud as they worked, and we recorded screen video and think-aloud audio for all participants.

8.4 Results

Our hypothesis for this work was:

> The P3P Expandable Grid allows website visitors to answer questions about a website's privacy policy more accurately and faster than does a natural-language based policy presentation.

The objective of our data analysis was to test this hypothesis. We also addressed the corollary hypothesis that the P3P Expandable Grid would lead to greater levels of satisfaction.

For the Web-based study, we measured accuracy rate and mean time-to-question-completion for each of the six comprehension questions and for each of the six conditions. Accuracy rate is the proportion of participants who correctly answered a comprehension question. Before computing accuracy rate and time-to-question-completion, we checked the data for participants who may have gamed the study by clicking as quickly as possible through the comprehension questions without putting effort into answering correctly. We eliminated from analysis those participants who finished faster than two standard deviations below the mean of the log-transformed time-to-question-completion data for three or more questions. We eliminated seven participants for this reason, including six Grid participants and one natural language participant.

In addition to accuracy rate and mean time-to-question-completion, we computed dropout rates, defined as the proportion of participants who started but did not complete the study in each condition, and scores on the subjective satisfaction questions.

For the lab-based study, we analyzed video and audio for evidence of usability problems with the Grid.

8.4.1 Accuracy results

We used logistic regression to test for an overall difference in accuracy rates across all six comprehension questions between the Grid (overall accuracy = 0.59) and the natural language (overall accuracy = 0.68) formats. We used format as the single factor in our logistic regression model. The model gave an intercept of 0.27 and a coefficient for format of 0.25. A Wald test of the hypothesis that the format coefficient was not equal to 0 was significant at the 0.05 level ($Z = 3.62, p < 0.001$), suggesting that there is an effect of format on accuracy. The sign of the coefficient, as well as the actual overall accuracy rates, suggests that over all questions, participants were more likely to correctly answer questions using the natural language format.

To follow up the logistic regression and gain a more detailed picture of how the Grid compared to the natural language format, we performed 18 pairwise comparisons of the Grid accuracy rate with the natural language accuracy rate for each question and for each policy length. We used one-sided Fisher's exact tests of the hypothesis that the Grid accuracy rates were higher than natural language accuracy rates. Since our experimental goal was to determine whether the Grid format was superior to the natural language format, we did not test the hypothesis that the natural language accuracy rates were higher than the Grid accuracy rates, even though, as it turns out, the natural language accuracy rates were in most cases higher. We corrected for multiple testing using the Benjamini-Hochberg method, which gave an adjusted per-test $alpha$ of 0.00008.

These pairwise comparisons showed that the Grid gave significantly higher accuracy rates for only one question, the SSN question, at the short and medium policy lengths. The results of our pairwise comparisons are shown in Table 8.1.

Table 8.1: *Summary of pairwise statistical tests for significant differences in accuracy rate for the P3P Expandable Grid (a_G) and accuracy rate for natural language (a_{NL}) for each comprehension question at each policy length. For all tests, the hypothesis tested was $a_G > a_{NL}$ against the null hypothesis $a_G = a_{NL}$. The p-values shown are from one-sided Fisher's exact tests; p-values below the α = 0.00008 adjusted rejection threshold are shown in bold, indicating significant tests.*

Task	Short			Medium			Long		
	a_G	a_{NL}	p	a_G	a_{NL}	p	a_G	a_{NL}	p
Cookies	0.90	0.99	1.00	0.92	0.98	0.99	0.96	0.99	0.96
Marketing Email	0.84	0.78	0.25	0.76	0.69	0.21	0.70	0.65	0.31
Opt-out	0.12	0.66	1.00	0.15	0.62	1.00	0.15	0.67	1.00
Share Email	0.21	0.47	1.00	0.17	0.54	1.00	0.44	0.54	0.93
SSN	0.56	0.27	$<$ **0.00008**	0.71	0.27	$<$ **0.00008**	0.66	0.49	0.02
Encryption	0.85	0.91	0.92	0.77	0.91	1.00	0.86	0.87	0.67

We could have used two-sided tests to simply determine whether Grid and natural language accuracy rates were different. Using two-sided tests could yield some statistically significant results in favor of natural language. We maintain that our one-sided analysis is appropriate and correct given our hypothesis that the Grid would lead to higher accuracy rates than natural language and our goal of developing better privacy policy presentations, rather than worse ones. However, it might be of interest to some readers to test the reverse hypothesis that the natural language accuracy rates are higher than Grid accuracy rates. We performed an alternative data analysis in which we used two-sided Fisher's exact tests for statistically significant differences between Grid and natural language accuracy rates. Because the hypotheses we were testing in our alternative analysis are not consistent with our *a priori* hypotheses, we used the Bonferroni multiple-testing correction, which is a very conservative correction (more conservative than the Benjamini-Hochberg method used in our primary analysis) that ensures that the probability of even one false positive from our testing is less than our chosen α of 0.05. The Bonferroni adjustment gives us a per-test α of 0.0028. Our alternative analysis shows the natural language accuracy rates for the Opt-out question for all policy lengths ($p < 0.001$ for short, medium, and long policies) and the Share email question for the short and medium policy lengths ($p < 0.001$ for the short and medium policies) to be statistically significantly higher than the Grid accuracy rates. Although it hardly seems interesting to demonstrate that the natural language format, which is known to be poor, is better than our mostly failed attempt to improve upon it, our data do support the hypothesis that natural language is a better format for some questions than the Grid.

8.4.2 Time-to-question-completion results

To test the hypothesis that the Grid format would lead to faster performance on comprehension questions, we computed mean time-to-question-completion for each question at each policy length and for each format. Mean time-to-question-completion for a comprehension question included completion times only for those participants who correctly answered the question, since we are interested in the time it takes to correctly comprehend a policy. We also excluded data points that were more than three standard deviations above or below the mean of the log-transformed time-to-question-completion data for any one question, policy length, and format. We assumed that such data points were the result of participants taking a break from the study in the middle of a question or rushing through the question, rather than merely the result of exceptionally slow or fast performance on the question. We excluded 11 data points (of 3078, which is 520 participants minus 7 eliminated for gaming the study times six questions), distributed across all six conditions.

An ANOVA comparing the overall hypothesis that the mean time-to-question-completion for all questions was different between the Grid ($M = 44.7, sd = 90.0$) and natural language ($M = 23.2, sd = 27.0$) formats was significant at the $\alpha = 0.05$ level ($F(1, 1966) = 31.055, p < 0.001$). The direction of the difference in means suggests the natural language format led to faster performance at completing comprehension questions, and the results of the ANOVA show this result to be statistically significant.

We followed up the ANOVA with 18 pairwise comparisons of the mean time-to-question-completion of each format for each question and policy length. We used one-sided t-tests to test the hypothesis that Grid mean time-to-question-completion was less than natural language mean time-to-question-completion. As with accuracy, we were not concerned with testing for significantly better performance by the natural language format since our experimental goal was to determine when the Grid format is an improvement on the natural language format. We corrected for multiple testing using the Benjamini-Hochberg method, which gave an adjusted per-test α of 0.004. Results of the pairwise comparisons are shown in Table 8.2.

8.4.3 Dropout rate results

We compared dropout rates between participants who were assigned to the Grid format and those who were assigned to the natural language format. We define dropout rate as the number of participants in a given condition who did not finish the comprehension questions divided by the number who started the study in that condition. In each condition, 393 participants started the study. In the Grid condition, 152 did not finish and in the Natural Language condition, 114 did not finish, so the Grid dropout rate was 152/393 = 38.7% and the natural language dropout rate was 114/393 = 29.1%. We tested for a statistically significant difference

Table 8.2: *Summary of statistical tests for significant differences in mean time-to-question-completion, in seconds, between successful Grid (G) and natural language (NL) participants for each comprehension question at each policy length. For each test, the table shows means (M) and standard deviations (sd) for each interface, and t-statistics (with degrees of freedom) and p-values resulting from one-sided t-tests; one p-value at the $\alpha = 0.004$ adjusted rejection threshold is shown in bold, indicating a significant test.*

	Short			Medium			Long		
Question	$M(sd)$	$t(df)$	p	$M(sd)$	$t(df)$	p	$M(sd)$	$t(df)$	p
Cookies	G:7(8) NL:9(6)	−2.6(154)	**0.004**	G:15(14) NL:9(7)	1.4(121)	0.92	G:14(14) NL:11(9)	0.4(145)	0.67
Marketing Email	G:21(25) NL:14(16)	1.0(120)	0.8	G:35(39) NL:15(14)	3.5(102)	1.00	G:31(61) NL:30(46)	−0.6(110)	0.27
Opt-out	G:81(79) NL:33(18)	3.2(10)	0.99	G:72(56) NL:36(20)	2.3(12)	0.98	G:72(70) NL:35(18)	1.7(12)	0.94
Share	G:53(43) NL:29(25)	1.1(23)	0.86	G:33(42) NL:26(22)	−0.3(15)	0.36	G:47(103) NL:35(45)	−0.3(62)	0.38
SSN	G:28(28) NL:17(16)	2.2(45)	0.98	G:39(43) NL:11(8)	4.4(66)	1.00	G:47(48) NL:18(15)	3.3(93)	1.00
Encryption	G:66(56) NL:22(15)	7.2(121)	1.00	G:102(127) NL:27(20)	6.8(98)	1.00	G:120(232) NL:44(55)	5.2(135)	1.00

in dropout rates using Fisher's exact test. We had no preconceived notion about the dropout rate, so we used a two-sided test. Fisher's exact test showed the difference in dropout rates to be significant at the $\alpha = 0.05$ level ($p = 0.005$). More participants dropped out of the Grid condition, and this result is statistically significant.

8.4.4 Subjective satisfaction results

We compared participants' subjective satisfaction with each presentation format. We first normalized participants' responses from all of the subjective satisfaction questions listed in section 8.3.1.4 by putting them on a scale from 1 to 7 where 1 indicates the most negative attitude and 7 indicates the most positive attitude. For example, a response of 7, "Strongly disagree," to the statement, "Finding information in Acme's privacy policy was a pleasurable experience" would receive a 1, the most negative rating, on our normalized scale, while a response of 7 to the statement, "It was hard to find information in Acme's policy" would remain a 7 on our normalized scale because of the negative wording of the question. After normalizing all responses, we took the mean response across all questions for participants who used the Grid ($M = 2.8, sd = 1.6$) and natural language ($mean = 3.8, sd = 1.5$). Two t-tests showed both the Grid ($t = -32.9(1899), p < 0.0001$) and natural language ($t = -5.8(2180), p < 0.0001$) means to be statistically significantly less than 4.0, the neutral score, at the $\alpha = 0.05$ level. This result suggests that participants in both formats had negative attitudes toward reading privacy policies. Another t-test showed the difference in means between the Grid and natural language formats to be significant at the $\alpha = 0.05$ level ($t = -20.9(3927), p < 0.0001$), suggesting attitudes were more positive toward the natural language format.

8.4.5 Lab-based study results

The objective of the data analysis for the lab-based study was to determine reasons why participants answered comprehension questions incorrectly using the Grid. To this end, we performed three analyses of the videos and think-aloud audio from the study. The three analyses together revealed eight usability problems that led to participants incorrectly answering comprehension questions. These analyses and usability problems are described in this section.

8.4.5.1 Analysis of where answers were found

In instances where participants incorrectly answered questions, we attempted to determine where in the Grid presentation the participant found the incorrect answer. It was sometimes possible based on what was on the screen, what participants had recently clicked on or scrolled to, and what participants were saying to pinpoint specific Grid squares, rows, or metadata from which they determined answers. Because this process was somewhat subjective, we used two raters and only included data from cases in which both raters agreed on where the participant had found an answer.

This analysis revealed four causes of incorrect answers. The causes were:

- *Multiple statements.* Answering comprehension questions for medium and long P3P policies was difficult because the policies contained multiple P3P statements. For the Cookies and Acme Email questions, the answers were only contained in a subset of the multiple statements, and for the Share Email and SSN questions, it was necessary to check all the statements. We observed five instances of participants finding an answer in the wrong statement or failing to check all statements.
- *Metadata.* Policy metadata that was out of the graphical region of the Grid was hard to find. Not surprisingly, participants seemed to focus on the graphical portion of the Grid presentation. In the Opt-out question, which required that participants read policy metadata, participants sometimes missed all or part of the necessary metadata to answer the question. We observed seven instances of such problems.
- *Terminology.* Participants generally seemed confused by P3P concepts and terminology. For example, we observed two instances of participants looking in the "Companies who help us" recipient column for the Share Email task, when they should have been looking in the "Other companies" recipient column. In the P3P policy, "Companies who help us" indicates companies who receive customer data to help complete orders but not for marketing purposes. "Other companies" indicates companies that might receive customer data for marketing purposes. Other examples P3P terms we found, informally, to be confusing included the "Opt-in" and "Opt-out" outcomes and the term "Profiling" to cover P3P's `pseudo-analysis`, `pseudo-decision`, `individual-analysis`, and `individual-decision` purposes, all of which are related to creating profiles of users.
- *Two dimensions, one axis.* Because P3P policies have three primary data-specific assertions, it really requires a three-dimensional table to represent a whole policy. In our design, we juxtaposed the recipient and purpose assertions together on one axis. This juxtaposition caused errors. In the Share Email question, participants had to find a row corresponding to "Email address" and then notice that the "Other companies" recipient column had a grey square. However, we observed two instances of participants thinking they found the answer in the "Marketing" purpose column. Had the purposes been separate from recipients, this error likely would not have happened.

8.4.5.2 Analysis of confusion statements.

We analyzed the think-aloud audio for instances in which participants expressed confusion. Two raters listened to the think-aloud audio and noted statements that, in their judgment, indicated confusion. Examples of such statements included:

- "The policy itself is not that – very clear at all."
- "Yeah, I'm just confused right now what I'm looking at. OK, so there's these color-coded things..."
- "I'm not sure how to find this, I'm just kind of looking around."

We took the union of all confusion statements identified by the two raters, identified three candidate problem categories that might be indicated by the statements, and then had two raters place the statements into the three categories. The two raters identified 30 total confusion statements. The statements were distributed across 11 of the 12 participants and five of the six tasks, so it does not appear that any one task or any one participant was responsible for a disproportionate number of the statements. When categorizing the statements, the raters agreed on categories for 21 of the 30 statements and agreed that six of the statements fit into none of the three categories. The raters disagreed on the categorization of three of the statements. Statements for which raters could not place into a category or for which they disagreed were eliminated from analysis.

Problem categories with example confusion statements include:

- *No focal point.* Perhaps the most common problem with the Grid was its lack of a clear place to start looking at it. The Grid contains a great deal of information, and can look visually cluttered. There is so much to see that participants often missed the information they needed to answer questions correctly. *Example confusion statement:* "This is a really convoluted presentation."
- *Multiple statements.* As we found in the previous analysis, participants were confused by multiple P3P statements. *Example confusion statement:* "There's no mention of credit cards at all, which is kind of strange, so I'm kind of playing the 'is it under this heading [referring to statement header boxes]' game again."
- *Confusing icons.* The Grid has fifteen distinct symbols that can be displayed in Grid squares, and participants sometimes did not understand what they all meant. *Example confusion statement:* "Yeah, I'm just confused right now what I'm looking at. OK, so there's these color-coded things..."

8.4.5.3 Analysis of Grid expansions.

Analyzing data from both the Web-based and lab-based studies on when participants expanded the Grid, we found two problems: first, that some participants seemed not to realize the Grid was interactive, and second, that many participants did not understand the P3P data hierarchy.

Non-expansion

We observed participants' expansions of the Grid's rows and columns. From the expansion data we recorded in the Web-based study, we observed that 35 of 241, or 14.5%, of Grid participants never expanded the Grid. In the lab-based study, we observed that one of 12 Grid participants never expanded the Grid. Some participants seem not to have realized the Grid is interactive.

Excess expansion

We informally observed that some participants were not familiar with the P3P data hierarchy and had to do quite a bit of searching to find some of the items they were looking for. For example, participants looking for "Social Security number" in the SSN task did not know that P3P has a "Social Security numbers and government IDs" category. They would scan the list of categories for one that seemed relevant to Social Security numbers, find "Social and economic categories," and click on that. Only later would they find what they were looking for under "Social Security numbers and government IDs," at the bottom of the list of data categories.

To quantitatively confirm that lack of understanding of the P3P hierarchy was problem, we counted the number of excess expansions of the P3P data hierarchy in the lab-based study. We defined an "excess expansion" the expansion of a Grid row or column that only revealed Grid squares that were irrelevant to answering the question at hand. Thus, if a participant expanded "Social and economic categories" during the SSN task, this expansion was an excess expansion. We observed 13 excess expansions distributed across five of the six questions and seven of the 12 participants.

8.5 Discussion

Our results strongly suggest that the P3P Expandable Grid, as currently implemented, is not an effective means for presenting privacy policies to Web users. Except in two cases (the SSN question and the Cookies question), participants using the Grid performed no better and no faster in correctly answering comprehension questions than participants using natural language. Moreover, the higher dropout rate amongst Grid participants suggests participants found the Grid more frustrating to use, and subjective satisfaction scores show participants strongly disliked the Grid.

Policy length did not have the effect we expected on our results. We expected the Grid to be a more scalable format than natural language, but, if anything, we witnessed the opposite effect. Multiple statements in the policy represented in the Grid led to the problem of checking the wrong statement or failing to check all relevant statements.

Our results show two bright spots for the Grid. First, participants using the Grid did perform better, as expected, on the SSN task, in which they were asked to determine that a website did *not* engage in a certain data practice. In real tasks, users may want to reassure themselves that a company does not engage in some objectionable practice. Second, participants answered the Cookies question faster for the short policy using the Grid. While we cannot conclusively state the reason for the faster performance on the Cookies task, informal observations from the lab-based study suggest some participants visually searched for the word "Cookies," and found it faster in the Grid because there were fewer words to search through.

However, our results are mostly negative with respect to the Grid, and this is somewhat surprising given how well the Expandable Grid concept has been shown to work in a user interface for another policy domain, file permissions [103]. Our lab-based study suggests eight reasons why the Grid was difficult to use:

1. *No focal point.* The Grid is visually busy, so it is hard to know where to start a visual search for information.
2. *Difficulty with hierarchy.* Users are not familiar with the P3P data hierarchy and do not know how to find relevant items in it.

3. *Multiple statements.* Policies with multiple P3P statements may lead users to find information in the wrong statement or to fail to check all relevant statements.
4. *Metadata.* Policy metadata is difficult to find.
5. *Confusing icons.* The meaning of some of the 15 possible icons is not apparent to some users.
6. *Terminology.* P3P privacy policy terminology is confusing and unfamiliar to many users.
7. *Two dimensions, one axis.* Juxtaposing two dimensions on one axis was confusing to users.
8. *Non-expansion.* Some users never realized the Grid is interactive and can be clicked to expand it.

8.5.1 Lessons for applying the Expandable Grid concept

The usability problems we observed suggest several lessons for how to design presentations of policies based on the Expandable Grid concept. We list these lessons in this section.

Provide a starting point such as a search bar.

The most common problem and probably the biggest impediment to finding answers in the Grid was the vast amount of visual information with no clear starting point. User interfaces using the Expandable Grid concept should provide a starting point; a search feature may serve as a starting point. A search bar was part of the successful file-permissions Expandable Grid design [103].

Provide a summary display.

The problems with lack of a focal point, difficulty with the P3P data hierarchy, and multiple statements could all be mitigated by simplifying the display of P3P policies. To this end, a simple summary display in which all statements are merged into one would be helpful. Also helpful would be a summary showing only data practices of the highest concern to consumers, such as sharing of health data or sharing contact information data for marketing purposes.

Use short labels.

The usability problems with terminology and metadata suggest some policies may not be well suited to display in an Expandable Grid. The problem of finding terminology to describe privacy concepts to Web users is known to be difficult. The P3P1.1 specification acknowledges the problem of finding concise, human-readable descriptions for privacy concepts and offers some such labels for P3P elements [37]. In our case, the problem was exacerbated by trying to find short terms that could be used as labels in the Grid. The Expandable Grid may be best suited to showing policies in which the concepts can be summarized with short labels on the axes (e.g., in file permissions, users have names that are generally around eight characters long and files have names that are usually less than 30 characters). We tried using short labels and provided longer descriptions when users moved their mouse over the labels, but participants in our lab study still seemed confused by the concepts. There might be better labels for P3P concepts than those we used, but natural language may be the most effective means for explaining nuanced policy concepts to users who have never encountered those concepts before. Natural language may also be more effective for presenting policies with large amounts of metadata that cannot be put into the matrix format required by the Grid.

Answer common questions in one place.

Multiple statements in P3P policies make it necessary sometimes to check multiple grid squares to find the answer to a simple, common question. Expandable Grid interfaces should be designed so that common questions can be answered by looking in one place. A summary view in which the contents of multiple statements are merged, as described above, could provide one place to look for answers to common questions.

Do not juxtapose two dimensions on one axis.

A tabular representation of a P3P policy requires three dimensions, one for each of the three P3P data-specific assertions. To fit three dimensions into a two-dimensional display, we tried juxtaposing two dimensions together on the same axis. The approach did not work. Some solution is needed to fit policies of more than two dimensions into a Grid presentation; while we do not know what solution is right, juxtaposing two dimensions together seems to be wrong. A better solution might be to require participants to choose a fixed cross-section of the policy to look at, e.g., to select "Marketing" as the purpose and then view a two-dimensional grid of data-by-recipient. It might be possible to display two dimensions on one axis by positioning a copy of one dimension's values under each value of the other dimension; we have yet to evaluate this approach.

Provide plenty of explanation for icons.

It is hard to help people understand a large number of icons. A legend helps, but may not be enough, since it draws the eyes away from the icons of interest. Help might be provided by showing the meaning of an icon when the mouse is moved over it. Also, reducing the number of icons needed to display a policy would help.

Emphasize or eliminate interactivity.

Users may have difficulty learning that a Grid presentation is interactive. We provided many cues of interactivity, including labels that read, "click for more," highlights on rows when the mouse was moved over them, and "+" symbols to indicate expandability. Still, many study participants did not realize the Grid is interactive. There might be even more indications we could provide to emphasize the Grid's interactivity, but another solution might be to eliminate interactivity altogether with a more compact policy representation that could fit on the screen without requiring expansion.

8.5.2 Limitations of our studies

We tested the P3P Expandable Grid on participants who had never seen the Grid before and had probably never encountered P3P before. While the Grid did not work well for displaying privacy policies to this class of users, it may be well suited to other use cases we did not test.

In particular, the Grid may work well as a authoring tool for P3P experts who want to write a P3P privacy policy. Indeed, our experience has been that we, as P3P experts, find it easier to understand the contents of a P3P policy by viewing it in the Grid's tabular form than in other common formats, such as a list of rules or raw XML. The Grid could be made interactive, so that authoring a P3P policy would be a simple matter of clicking on Grid squares to edit the policy. P3P policy authoring is currently done in raw XML or using somewhat unwieldy tools like the IBM P3P Policy Editor [40].

Another use case to which the Grid may be better suited is as a standardized privacy policy presentation. Participants in our study had never seen the Grid format before. If Web users saw privacy policies presented in a consistent format, such as the Grid, repeatedly across all websites, they might become more familiar with that format over time and ultimately find it easy to read. A standardized format would also help users compare privacy policies between websites. In contrast, comparing natural language policies across websites can be extraordinarily difficult.

We tested the P3P Expandable Grid presentation format against the natural language presentation format for a single organization. We believe the organization we chose may have an exceptionally well-written natural language privacy policy. The Grid may perform better against average or poorly written privacy policies.

8.6 Conclusion

We developed a means for graphically presenting website P3P privacy policies based on the Expandable Grid concept and found that it did not generally improve Web users' comprehension of privacy policies compared to a natural language policy presentation. However, the Grid did perform well at indicating practices *not* allowed by a policy, and it may hold promise for allowing P3P experts to navigate and author policies. A simplified Grid presentation of P3P policies may be an effective means for presenting policies even to general Web users in a standardized format. In future work, we will continue efforts toward designing such a simplified P3P policy presentation.

CHAPTER 9

Other Applications of the Expandable Grid

The Expandable Grid concept has been applied to displaying policies in other domains besides Windows XP file permissions and website P3P privacy policies, although we have not tested it in these other domains. Here, we describe applications of the Grid concept to three other domains: Grey, a distributed building access-control system; Perspective, a distributed data storage system; and PeopleFinder, a location-disclosure system. These applications collectively show the potential versatility of the Expandable Grid concept for displaying policies in a variety of domains.

9.1 Grey

Grey is a distributed access-control system that allows people to use smartphones to control access to offices, perimeter doors, other locked spaces, and computers in an office environment [18]. Grey has been deployed to over 30 users in our university office building. Grey users who own resources, such as their offices, can delegate access to those resources using their smartphone. In delegating access to others, Grey resource owners create access-control policies [16]. We have designed and implemented an Expandable-Grid-based user interface, the Grey Expandable Grid, for viewing and editing these access-control policies on a desktop computer. A screenshot of our interface is shown in Figure 9.1.

The Grey Expandable Grid illustrates how we can display time as a policy attribute in an Expandable Grid while still showing principals and resources. In the Grey Expandable Grid, time is shown on the vertical axis on the left and resources and principals are combined in the horizontal axis along the top. A dark horizontal line in the grid itself shows the current time, with past policy shown above the line and future policy shown below. Time is shown in a sliding window that can be both zoomed (i.e., the window size can be adjusted) and scrolled (i.e., the window can be slid) using the scrollbar on the right. We adapted this approach to showing time from the DateLens system [19]. Resources are shown as the primary dimension along the horizontal axis, with a copy of the entire structure of principals shown beneath each resource. Both resources and principals can be grouped; "CUPS" is a group of resources that contains the resources "Cube,"

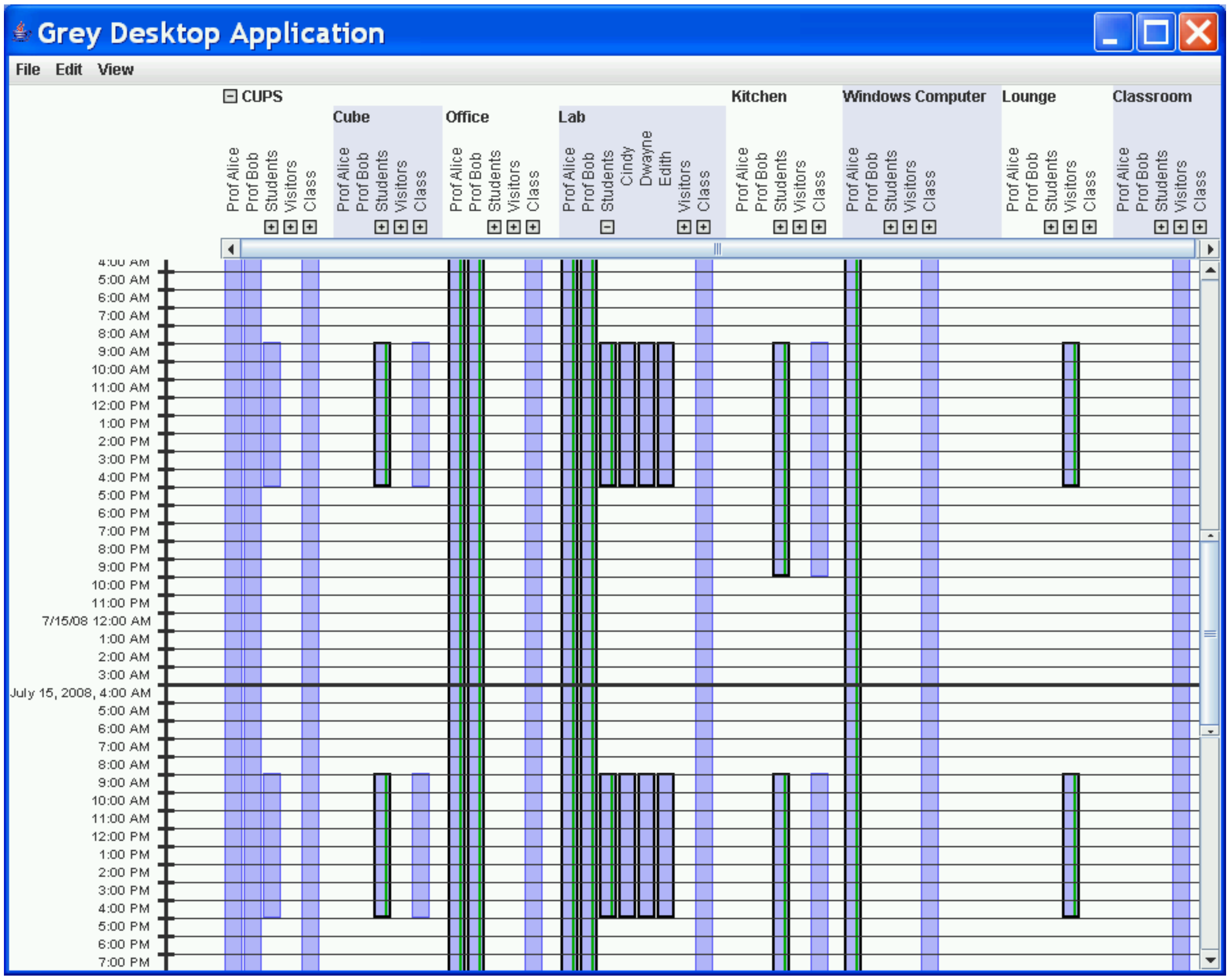


Figure 9.1: *A screenshot of our Grey Expandable Grid interface. The Grey Expandable Grid shows how policies with a time attribute can be shown in an Expandable Grid.*

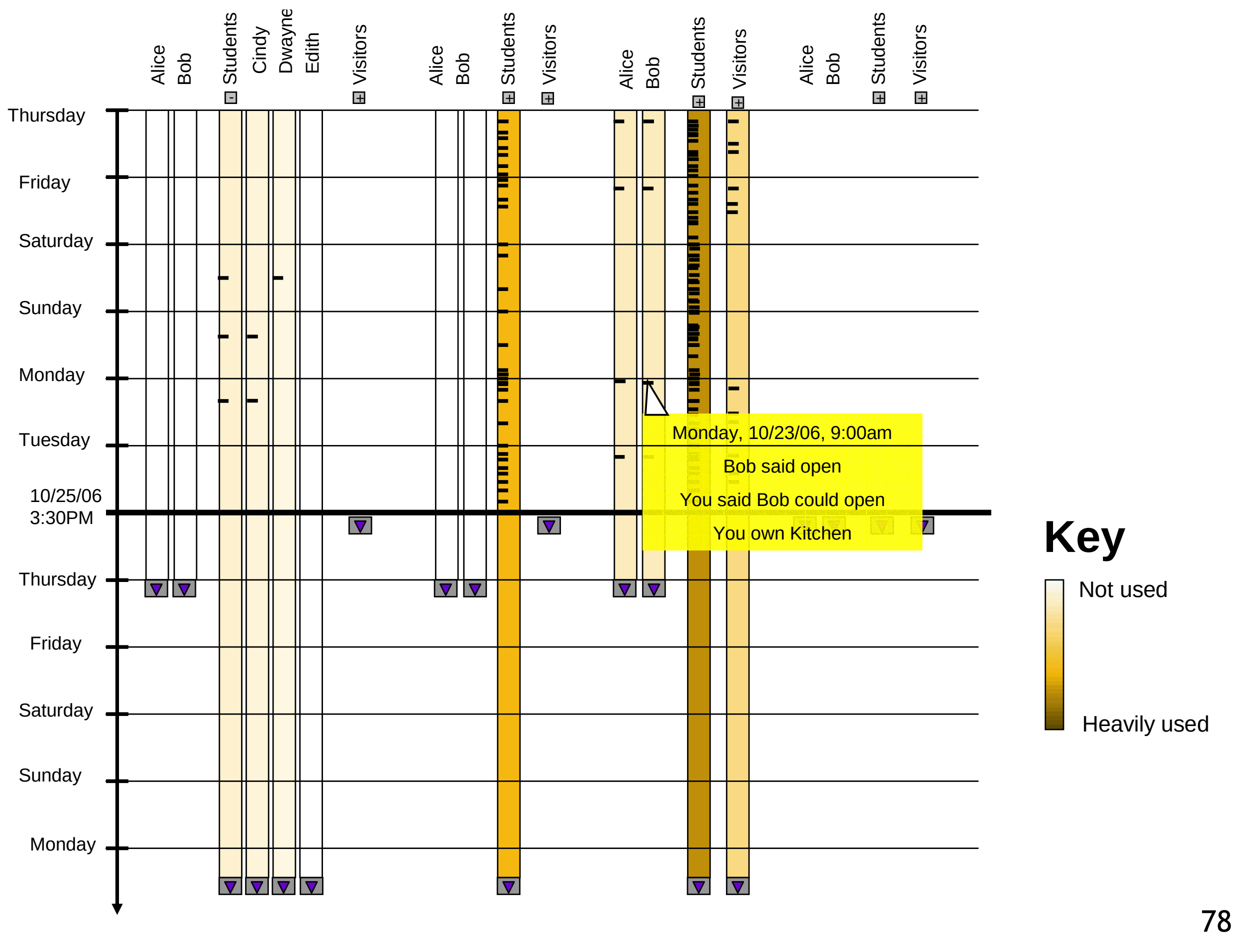


Figure 9.2: *An early mockup of our Grey Expandable Grid interface showing past access attempts as tick marks above the dark horizontal line that represents the current time. The past access data would allow a resource owner to audit past accesses.*

"Office," and "Lab." "Students" is a group of principals that contains "Cindy," "Dwayne," and "Edith." (The expanded version of the "Students" group can be seen beneath the "Lab" resource in Figure 9.1.)

In the grid, policy is shown using blue-shaded bars. A blue bar in the column corresponding to a resource and principal indicates the principal is (or was) able to access the resource for the amount of time the bar spans vertically. For example, in Figure 9.1, we see that Students were able to access the Lab from 9am to 5pm yesterday and will be able to access the lab from 9am to 5pm today. Bars with a thick border indicate cases where all principals represented by the bar have access to all resources represented by the bar. Bars with a light border serve the same function as yellow squares in the file-permissions Expandable Grid; they indicate cases where some of the principals have access to some of the resources represented by the bar. Green bands on bars indicate access that was explicitly granted through an access rule. Bars without green bands indicate access that is inherited from a group.

Figure 9.2 shows an early mockup of the Grey Expandable Grid interface in which we included features for helping resource owners audit past accesses to their resources. Tick marks in the bars above the dark horizontal line indicate past access attempts. By mousing over each tick mark, a resource owner could get specific data about each past access and why it was allowed or denied.

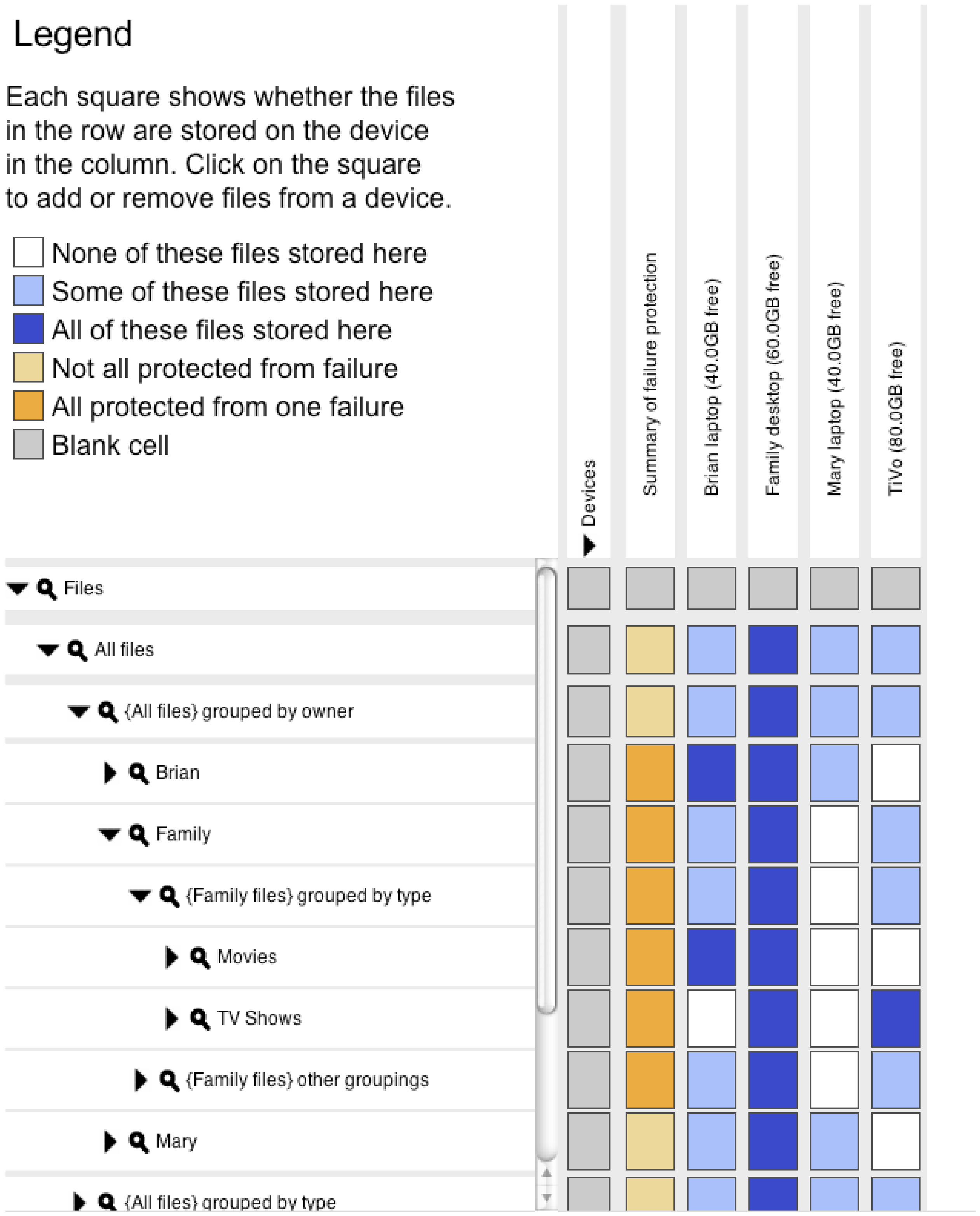
Although our Grey Expandable Grid design shows how a time attribute could be shown in an Expandable Grid, we have not yet deployed the Grey Expandable Grid or tested it in user studies.

9.2 Perspective

Perspective is a data storage system targeted toward non-technical home users [112]. It is intended to help users store, find, and manage their files by using a semantic file system, in which resources are organized in intuitive, flexible structures according to metadata tags, as opposed to a strict file hierarchy. Perspective also provides convenient distributed storage of resources across the multiple devices that may be found in the home, such as desktops, laptops, and digital video recorders.

Salmon et al. used the Expandable Grid to design a user interface for managing data storage policies in Perspective [111]. Their interface is shown in Figure 9.3. Resources, grouped into collections according to metadata, are shown on the vertical axis and devices are shown along the horizontal axis. The grid squares show which resources are stored on which devices. Perspective users can use the Grid interface to specify where they would like a given resource or collection of resources stored.

Salmon et al. ran a user study to test the usability of the Grid for data storage management in Perspective [111]. Although they designed their study more to test their architecture than the Grid interface, they did make some observations relevant to the interface. They found, as we did in our study on the P3P Expandable Grid, that the large amount of information contained in the Grid was overwhelming and confused participants at first. However, they report that after completing several tasks, participants "felt quite comfortable" with the Grid. Salmon et al. report a higher-than-expected overall accuracy rate for study tasks, so the combination of the Grid interface with the Perspective architecture seems to have been very effective. The Perspective Grid interface shows that the Expandable Grid concept is not restricted to applications in security and privacy policies, but is general enough to apply to policy management in a completely different domain.



Legend

Each square shows whether the files in the row are stored on the device in the column. Click on the square to add or remove files from a device.

⬜	None of these files stored here
▧	Some of these files stored here
🟦	All of these files stored here
🟨	Not all protected from failure
🟧	All protected from one failure
▨	Blank cell

	Devices	Summary of failure protection	Brian laptop (40.0GB free)	Family desktop (60.0GB free)	Mary laptop (40.0GB free)	TiVo (80.0GB free)
Files	▨	▨	▨	▨	▨	▨
All files	▨	🟨	▧	🟦	▧	▧
{All files} grouped by owner	▨	🟨	▧	🟦	▧	▧
Brian	▨	🟧	🟦	🟦	▧	⬜
Family	▨	🟧	▧	🟦	⬜	▧
{Family files} grouped by type	▨	🟧	▧	🟦	⬜	▧
Movies	▨	🟧	🟦	🟦	⬜	⬜
TV Shows	▨	🟧	⬜	🟦	⬜	🟦
{Family files} other groupings	▨	🟧	▧	🟦	⬜	▧
Mary	▨	🟨	▧	🟦	▧	⬜
{All files} grouped by type	▨	🟨	▧	🟦	▧	▧

Figure 9.3: *An Expandable-Grid-based user interface for data storage management in the Perspective data storage system. Image taken from Salmon et al. [111].*

9.3 PeopleFinder

PeopleFinder is a location-disclosure application that tracks people's locations via their cell phones and laptops [110]. PeopleFinder enables people to specify location-disclosure policies that express who can see their location at what times and under what circumstances. We have designed an Expandable-Grid-based interface for viewing and editing PeopleFinder policies. A mockup (by Madhu Prabaker) of the interface can be seen in Figure 9.4. Like the Grey Expandable Grid, the PeopleFinder Grid interface shows time along one axis, the horizontal in this case. The PeopleFinder Grid interface, however, is restricted to showing one week of time; in the current system, PeopleFinder policies can only be set to times within a generic week, not corresponding to specific dates. Principals and groups of principals are shown along the vertical axis. The grid uses colored bars to show the times each principal is allowed access to the policy author's location. As shown in the mockup, the interface could also be used for auditing by showing past access attempts as hash marks in the grid.

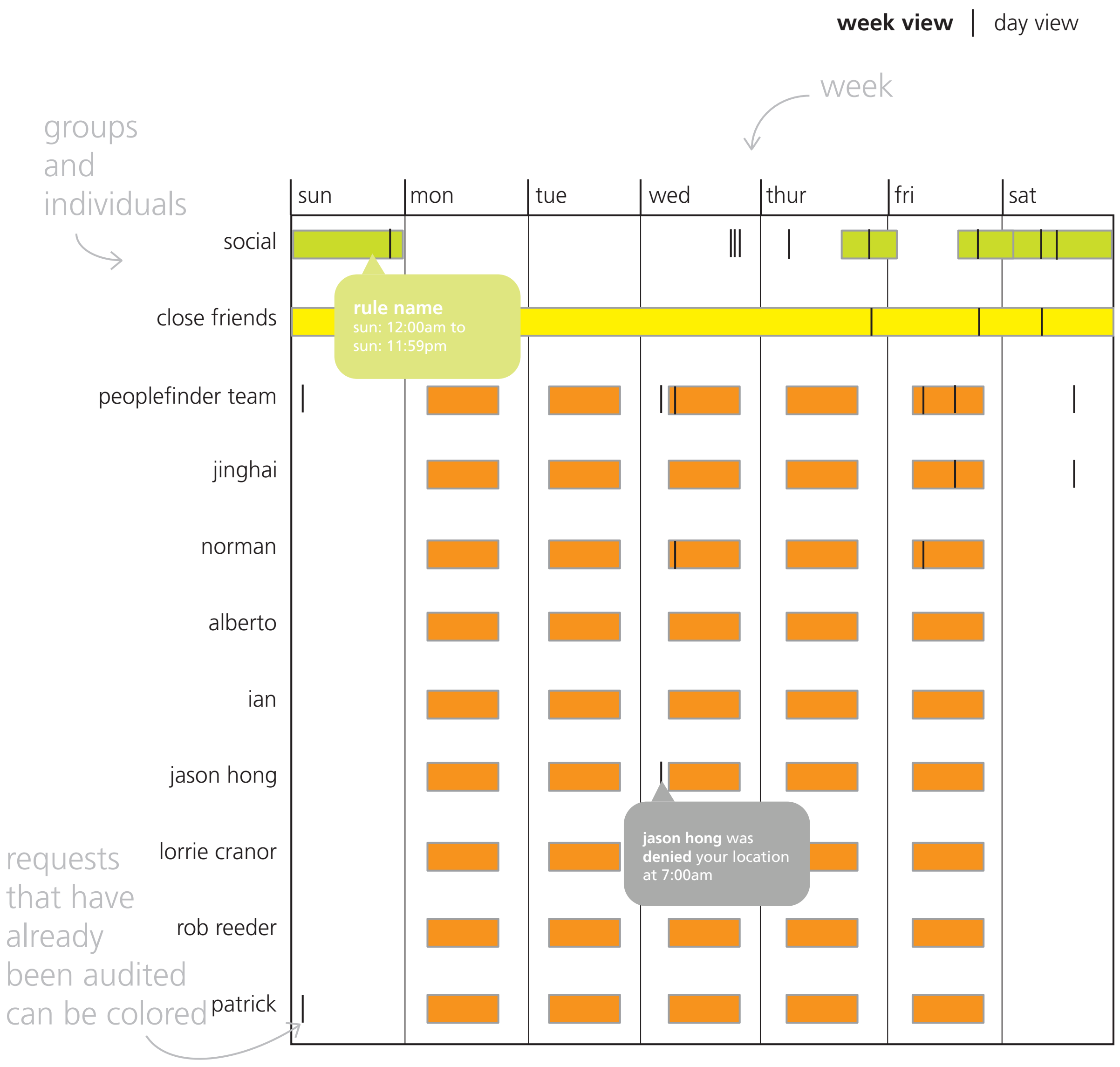


Figure 9.4: *A mockup (by Madhu Prabaker) of an Expandable-Grid-based user interface for viewing and editing PeopleFinder location-disclosure policies.*

PART IV

Conclusion

CHAPTER 10

Conclusions and Future Work

This thesis has addressed policy-authoring interface usability. We have shown that today's list-of-rules paradigm for designing policy-authoring interfaces leads to error-inducing interfaces that do not support policy authors' needs. We have argued that the primary reason the list-of-rules paradigm leads to errors is that it does not show effective policy to authors, but instead forces authors to translate access rules into effective policy by applying obscure default and conflict-resolution rules in their heads. To improve the state of policy-authoring interface design, we have presented the concept of the Expandable Grid, an interactive matrix visualization of a policy. We applied the concept to the design of user interfaces for two policy domains: Windows file permissions and website P3P privacy policies. We then tested the hypothesis:

> **Policy-authoring user interfaces using techniques centered on visualization of a full, effective policy lead to faster, more accurate performance on policy-authoring and policy-viewing tasks than do today's list-of-rules and natural-language policy-authoring interfaces.**

To test the hypothesis, we conducted user studies to evaluate the Expandable-Grid-based interfaces we developed for file permissions and P3P. We found that the file-permissions Expandable-Grid interface significantly improved performance across a broad range of policy-authoring tasks compared to the Windows list-of-rules interface. In contrast to this result, we found that the P3P Expandable Grid, for the most part, did not improve performance on policy-viewing tasks. We identified reasons for the Grid's disappointing performance and guidelines for how it should be applied. The hypothesis is partially supported by our work; interfaces centered on the Expandable Grid's visualization of full, effective policy *can* lead to faster, more accurate performance on policy-authoring and viewing tasks, but only if applied carefully. We conclude that the Expandable Grid concept is a promising approach to policy-authoring interface design across a wide variety of policy domains if it is applied according to the lessons we have learned.

10.1 Contributions

The original contributions of this thesis provide guidance for the design of usable policy-authoring interfaces. The original contributions are:

- *Key insight: show effective policy*: If there is one single message to take away from our work, it is that policy-authoring interfaces should prominently show effective policy. Prominently showing effective policy supports viewing the policy; to support changing policy, policy-authoring interfaces should, to the extent possible, allow authors to directly manipulate the effective policy, rather than forcing them to manage an arcane set of rules.

- *Expandable Grid concept*: The Expandable Grid concept can be applied to a wide variety of policy domains. It provides a scalable means for viewing the full, effective policy.

- *Expandable Grid implementation for file permissions*: We have shown how the Expandable Grid concept can be used in the design and implementation of an actual user interface for setting file permissions.

- *Evaluation of the file-permissions Expandable Grid*: Our evaluation of the file-permissions Expandable Grid interface has shown that applying the Expandable Grid concept can greatly improve performance on a broad range of policy-authoring tasks compared to a list-of-rules interface. Our results suggest the reason for the Grid's performance superiority are its display of the full, effective policy.

- *Policy semantics for the file-permissions Expandable Grid*: We have described a semantics for a file-permissions access-control system that is well suited to use with an Expandable Grid. We have shown that a semantics can have a significant effect on usability. We have argued that our semantics, which is based on specificity precedence in both the principal and resource dimensions, can lead to usability gains compared to the Windows access-control semantics by staying more faithful to the properties of direct manipulation, exception-rule preservation, order independence, and fail safety.

- *Expandable Grid implementation for P3P*: We have applied the Expandable Grid concept to the domain of website P3P policies. We implemented an interface that presents P3P policies to website visitors in a consistent, standardized, graphical format.

- *Evaluation of the P3P Expandable Grid*: Our evaluation of the P3P Expandable Grid showed that, for the most part, it did not improve performance on policy-viewing tasks compared to a natural language policy presentation. Our evaluation revealed several reasons why we did not observe the expected improvement.

- *Lessons for Expandable Grid application*: Our evaluation of the P3P Expandable Grid suggested lessons for how to apply the Expandable Grid concept successfully. These lessons, described in detail in Chapter 8 are:
  - Provide a starting point such as a search bar;
  - Provide a summary display;
  - Use short labels;

  - Answer common questions in one place;
  - Do not juxtapose two dimensions on one axis;
  - Provide plenty of explanation for icons; and
  - Emphasize or eliminate interactivity.

10.2 Future work

Future work falls into two categories. First, Expandable Grid interfaces could potentially benefit from a variety of features we have not yet explored. We list several of these potentially beneficial features here. Second, our work in evaluating Expandable Grid interfaces has been limited to specific classes of policy authors or policy consumers. We discuss a line of future work in developing policy-authoring interfaces for a broader audience than the audience we considered in this thesis.

10.2.1 Improving the Expandable Grid

Many potential features could improve the usability of the Expandable Grid or allow it to be applied more broadly. These features come largely from the information visualization literature and we have already alluded to some of them in Section 5.3.2. We list some of these features here.

- *Distortion*: Our implementations of the Expandable Grid concept are quite limited in the amount of a policy they can show on one screen without scrolling. Distortion could be employed to allow more policy to fit on one screen [33, 52, 80, 116, 117]. In a Grid using distortion, rows of interest (such as the row the mouse is pointing to) would be given more display height than other rows; columns of interest would be given more display width than other columns [101]. Rows and columns not deemed to be of interest would be given as little as one display pixel of height or width, thus allowing potentially hundreds of rows and columns to be visible in low detail and a few to be visible in full detail at one time. Geometrically, the distortion can be done in a variety of ways, as documented in Leung and Apperley [80] and Carpendale et al. [34].

- *More than two attributes*: Policies often have more than two attributes, so showing them in a two-dimensional space requires special techniques. We have discussed some of our solutions to the more-than-two-attributes problem in Section 5.3.2.3, including subgrids and juxtaposition. Other methods that could be applied to the Expandable Grid include slicing (showing a cross section of a policy by setting fixed values for attributes not displayed) [63, 133], various forms of aggregation [33, 63], placing grids within grids [8, 46], and zooming [20].

- *Brushing*: Brushing describes interactive techniques for selecting a set of items on which an operation is to be performed [142, 106]. For example, using the mouse to draw a rectangle around a group of grid cells would be an example of a brushing technique. The interesting aspect of brushing is that it enables executing an operation on a collection of objects, instead of only executing operations on one object at a time. Brushing might be employed in the Grid to enable changing multiple policy settings at a time.

- *Filtering*: Our Expandable Grid interfaces show a great deal of information, but authors may want to limit the amount that is shown at one time. Filtering is one technique for limiting the amount of information shown in a visualization [5, 49]. An Expandable Grid interface might use filtering to allow an author to show only a relevant subset of a policy. For example, a file-permissions policy author might want to see only users in a certain group or only files with modification dates within a certain window.

- *Displaying time*: Since many policies have time as an attribute, it would be useful to develop a way to show time in the Expandable Grid. In our implementation of an Expandable Grid interface for Grey building access-control policies described in Chapter 9, we adopted Bederson et al.'s approach from their DateLens system [19], but we have not evaluated the approach in user studies.

- *Sorting by color*: Since policies are often sparse (i.e., in access-control policies, most decisions in a policy are DENY, with only a few ALLOW decisions), it may be desirable to collect all of the interesting parts of a policy (e.g., the ALLOWdecisions in an access-control policy) in the same place. We have implemented sort-by-color functionality in our file-permissions Expandable Grid, but we leave evaluation of the feature to future work.

- *Details on demand*: It would be useful to be able to click on a cell in the Expandable Grid to get more information about the policy data represented in that cell. Perhaps a dialog box could pop up to show all the policy values and the decision represented by that cell plus an annotation about why the policy was set the way it was. This idea is what Shneiderman calls "details on demand," and it is a common feature in information visualization interfaces [122].

- *Label rotation*: As mentioned in Section 5.3.2.4, displaying column labels in the Expandable Grid is a problem. Vertical labels are hard to read, but horizontal labels take up too much space. A variety of techniques could be used to mitigate this problem. We have ongoing work in one such technique, rotating text to horizontal when it is moused over.

- *Static checking for conflicts*: In Section 7.1.3, we mentioned static checking as one means for resolving rule conflicts. With static checking, a policy-authoring application could analyze the rule base for conflicts any time a rule is added or changed. If an added or changed rule causes a conflict, the policy could be alerted immediately and required to change the policy so that the conflict goes away or specify how the author would like the conflict to be resolved.

10.2.2 Usable policy authoring for all

Our conclusions about the usability and applicability of the Expandable Grid concept and a policy semantics are limited to specific classes of policy authors and viewers. Participants in our user studies of the file-permissions Expandable Grid interface and our specificity semantics were technically trained but non-expert policy authors. These policy authors are an important class; they represent the kind of novice or occasional policy authors we intended to design for. They are similar to novice system administrators like the administrator involved in the Memogate scandal [132] or some of those studied by Barrett et al., who report varying levels of skill and expertise amongst system administrators [13]. Nevertheless, the policy authors we studied do not represent everyone who authors permissions policies. Our participants can be thought of as falling in the middle of a spectrum of policy-authoring skill. Authors at the extreme ends of the spectrum may offer interesting challenges for policy-authoring interface design. At one end of the spectrum are novice, non-technical authors. Supporting this class of authors matters more and more as we move into a computing world where every individual stores and shares personal information online and may want to restrict access to that information. However, the Expandable Grid may simply show too much information for this class of authors to handle. It may be that lists of rules or natural language policy presentations are better suited to these novice policy authors. At the other end of the spectrum are highly experienced system administrators. This class of policy authors may be suspicious of graphical user interfaces for system administration tasks [128], so may be inclined to dislike an Expandable-Grid-based interface. In any case, system administrators may manage systems with thousands of users and millions of resources, so they need scripting and automation tools that scale to manage these extremely large policies. We believe the Expandable Grid may have something to offer expert system administrators, but that it would require extensive work to make it scale so that it is a convenient and efficient tool for viewing and managing very large policies. In future work, we will elicit the needs of these two classes of policy authors—non-technical and highly experienced—and design tools, possibly based on the Expandable Grid concept, to meet those needs.

Participants in our user studies of the P3P Expandable Grid were ordinary Web users. The Expandable Grid presentation we gave them was overwhelming to many of them. In future work, we will design a simpler website privacy policy presentation, possibly based on the Expandable Grid concept but with a simpler layout and less visual clutter. Although the P3P Expandable Grid did not support these novice policy viewers, we believe that even in its current form it may be very useful to a more advanced class of users, namely, expert P3P policy authors. In future work, we will test the P3P Expandable Grid for its ability to support P3P experts.

Bibliography

[1] A. Adams and M. A. Sasse. Users are not the enemy. *Communications of the ACM*, 42(12):41–46, December 1999.

[2] D. Agrawal, J. Giles, K.-W. Lee, and J. Lobo. Policy ratification. In *Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2005)*, pages 223–232, Los Alamitos, CA, June 2005. IEEE Computer Society Press.

[3] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. An XPath-based preference language for P3P. In *Proceedings of the 12th International Conference on World Wide Web*, pages 629–639, New York, NY, 2003. ACM Press.

[4] A. Agresti. *Categorical Data Analysis*. Wiley-Interscience, New York, NY, 2002.

[5] C. Ahlberg and B. Shneiderman. Visual information seeking: Tight coupling of dynamic query filters with starfield displays. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems(CHI 1994)*, pages 313–317, New York, NY, 1994. ACM Press.

[6] E. S. Al-Shaer and H. H. Hamed. Firewall Policy Advisor for anomaly discovery and rule editing. In *Proceedings of the IFIP/IEEE Eighth International Symposium on Integrated Network Management*, IFIP International Federation for Information Processing, pages 17–30, New York, NY, March 2003. Springer.

[7] A. I. Anton, J. B. Earp, Q. He, W. Stufflebeam, D. Bolchini, and C. Jensen. Financial privacy policies and the need for standardization. *IEEE Security and Privacy*, 2(2):36–45, March/April 2004.

[8] V. Anupam, S. Dar, T. Leibfried, and E. Petajan. Dataspace: 3-d visualizations of large databases. In *1995 IEEE Symposium on Information Visualization (InfoVis 1995)*, pages 82–88, Washington, DC, 1995. IEEE Computer Society.

[9] Apple Computer. Mac OS X Server: Workgroup Manager. Technology Brief available from `http://www.apple.com/lae/server/macosx/workgroup.html`, 2005. Accessed on June 8, 2008.

[10] P. Ashley, S. Hada, G. Karjoth, C. Powers, and M. Schunter. Enterprise Privacy Architecture Language (EPAL 1.2). W3C Member Submission 10-Nov-2003, November 2003. Available at http://www.w3.org/Submission/EPAL.

[11] D. Balfanz. Usable access control for the World Wide Web. In *Proceedings of the 19th Annual Computer Security Applications Conference*, pages 406–415, Los Alamitos, CA, December 2003. IEEE Computer Society. Available at http://www.acsac.org/2003/papers/43.pdf.

[12] D. H. Ballard, M. H. Hayhoe, and J. B. Pelz. Memory representations in natural tasks. *Journal of Cognitive Neuroscience*, 7(1):66–80, 1995.

[13] R. Barrett, E. Kandogan, P. P. Maglio, E. M. Haber, L. A. Takayama, and M. Prabaker. Field studies of computer system administrators: analysis of system management tools and practices. In *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work (CSCW 2004)*, pages 388–395, New York, NY, 2004. ACM Press.

[14] Y. Bartal, A. Mayer, K. Nissim, and A. Wool. Firmato: A novel firewall management toolkit. *ACM Transactions on Computer Systems*, 22(4):381–420, November 2004.

[15] Y. Bartal, A. Mayer, and A. Wool. Firmato: A novel firewall management toolkit. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy (S&P 1999)*, pages 17–31, Los Alamitos, CA, May 1999. IEEE Computer Society Press.

[16] L. Bauer, L. F. Cranor, R. W. Reeder, M. K. Reiter, and K. Vaniea. A user study of policy creation in a flexible access-control system. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems(CHI 2008)*, pages 543–552, New York, NY, April 2008. ACM Press.

[17] L. Bauer, L. F. Cranor, M. K. Reiter, and K. Vaniea. Lessons learned from the deployment of a smartphone-based access-control system. In *Proceedings of the 2007 Symposium on Usable Privacy and Security (SOUPS 2007)*, pages 64–75, New York, NY, July 2007. ACM Press.

[18] L. Bauer, S. Garriss, J. M. McCune, M. K. Reiter, J. Rouse, and P. Rutenbar. Device-enabled authorization in the grey system. Technical Report CMU-CS-05-111, Carnegie Mellon University, Pittsburgh, PA, February 2005. Available at http://reports-archive.adm.cs.cmu.edu/anon/2005/CMU-CS-05-111.pdf.

[19] B. B. Bederson, A. Clamage, M. P. Czerwinski, and G. G. Robertson. DateLens: A fisheye calendar interface for PDAs. *ACM Transactions on Computer-Human Interaction*, 11(1):90–119, March 2004.

[20] B. B. Bederson, J. D. Hollan, K. Perlin, J. Meter, D. Bacon, and G. Furnas. Pad++: A zoomable graphical sketchpad for exploring alternate interface physics. *Journal of Visual Languages and Computing*, 7(1):3–32, March 1996.

[21] D. Bell and L. LaPadula. Secure computer system: Unified exposition and Multics interpretation. Technical Report MTR-2997, MITRE Corporation, Bedford, MA, March 1976. Available at `http://csrc.nist.gov/publications/history/bell76.pdf`. Accessed on June 21, 2008.

[22] D. Besnard and B. Arief. Computer security impaired by legitimate users. *Computers & Security*, 23(3):253–264, May 2004.

[23] R. Bobba, S. Gavrila, V. Gligor, H. Khurana, and R. Koleva. Administering access control in dynamic coalitions. In *Proceedings of the 19th Large Installation System Administration Conference (LISA '05)*, pages 249–261, Berkeley, CA, December 2005. USENIX Association.

[24] D. F. C. Brewer and M. J. Nash. The chinese wall security policy. In *Proceedings of the 1989 IEEE Symposium on Security and Privacy*, pages 206–214, Los Alamitos, CA, May 1989. IEEE.

[25] C. Brodie, C.-M. Karat, and J. Karat. An empirical study of natural language parsing of privacy policy rules using the SPARCLE policy workbench. In *Proceedings of the 2006 Symposium on Usable Privacy and Security (SOUPS 2006)*, pages 8–19, New York, NY, July 2006. ACM Press.

[26] C. Brodie, C.-M. Karat, J. Karat, and J. Feng. Usable security and privacy: a case study of developing privacy management tools. In *Proceedings of the 2005 Symposium on Usable Privacy and Security (SOUPS 2005)*, pages 35–43, New York, NY, July 2005. ACM Press.

[27] M. Brown and R. Muchira. Taxonomy of conflicts in network security policies. *IEEE Communications*, 5(1):62–70, March 2004.

[28] M. D. Byrne. Reading vertical text: Rotated vs. marquee. In *Proceedings of the Human Factors and Ergonomics Society 46th Annual Meeting*, pages 1633–1635, Santa Monica, CA, 2002. Human Factors and Ergonomics Society.

[29] X. Cao and L. Iverson. Intentional access management: Making access control usable for end-users. In *Proceedings of the Second Symposium on Usable Privacy and Security (SOUPS 2006)*, pages 20–31, New York, NY, 2006. ACM Press.

[30] S. Card. Information visualization. In J. A. Jacko and A. Sears, editors, *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications*, chapter 28, pages 544–582. Lawrence Erlbaum Associates, Mahwah, NJ, 2003.

[31] S. K. Card, J. D. Mackinlay, and B. Shneiderman. *Readings in Information Visualization: Using Vision to Think*, chapter Information Visualization, pages 306–309. Morgan Kaufmann, San Francisco, CA, January 1999.

[32] S. K. Card, J. D. Mackinlay, and B. Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann, San Francisco, CA, January 1999.

[33] S. K. Card, J. D. Mackinlay, and B. Shneiderman. *Readings in Information Visualization: Using Vision to Think*, chapter Focus + Context, pages 306–309. Morgan Kaufmann, San Francisco, CA, January 1999.

[34] M. S. T. Carpendale, D. J. Cowperthwaite, and F. D. Fracchia. Extending distortion viewing from 2D to 3D. *IEEE Computer Graphics and Applications*, 17(4):42–51, July-August 1997.

[35] Center for Information Policy Leadership. Ten steps to develop a multilayered privacy policy, 2007. Available at http://www.hunton.com/files/tbl_s47Details%5CFileUpload265%5C1405%5CTen_Steps_whitepaper.pdf. Accessed on May 19, 2008.

[36] G. Conti. *Security Data Visualization*. No Starch Press, San Francisco, CA, 2007.

[37] L. Cranor, B. Dobbs, S. Egelman, G. Hogben, J. Humphrey, M. Schunter, D. A. Stampley, and R. Wenning. The Platform for Privacy Preferences 1.1 (P3P1.1) specification. W3C Recommendation, November 2006. Available at http://www.w3.org/TR/P3P11/. Accessed on May 19, 2008.

[38] L. Cranor, S. Egelman, S. Sheng, A. M. McDonald, and A. Chowdhury. P3P deployment on websites. *Electronic Commerce Research and Applications*, 50, 2008. To appear. Available at http://lorrie.cranor.org/pubs/p3p-deployment.pdf. Accessed on February 26, 2008.

[39] L. Cranor, M. Langheinrich, M. Marchiori, M. Presler-Marshall, and J. Reagle. The Platform for Privacy Preferences 1.0 (P3P1.0) specification. W3C Recommendation, April 2002. Available at http://www.w3.org/TR/P3P/. Accessed on May 19, 2008.

[40] L. F. Cranor. *Web Privacy with P3P*. O'Reilly, Sebastopol, CA, 2002.

[41] L. F. Cranor, P. Guduru, and M. Arjula. User interfaces for privacy agents. *ACM Transactions on Computer-Human Interaction*, 13(2):135–178, June 2006.

[42] P. Dewan and H. Shen. Controlling access in multiuser interfaces. *ACM Transactions on Computer-Human Interaction*, 5(1):34–62, March 1998.

[43] D. J. Dougherty, K. Fisler, and S. Krishnamurthi. Specifying and reasoning about dynamic access-control policies. In *Proceedings of the Third International Joint Conference on Automated Reasoning (IJCAR 2006)*, Lecture Notes in Computer Science, Vol. 4130, pages 632–646, New York, NY, August 2006. Springer.

[44] K. A. Ericsson and H. A. Simon. *Protocol Analysis: Verbal Reports as Data*. MIT Press, Cambridge, MA, revised edition, 1993.

[45] Federal Trade Commission. Privacy online: A report to congress, June 1998. Available at http://www.ftc.gov/reports/privacy3/priv-23a.pdf. Accessed on February 26, 2008.

[46] S. Feiner and C. Beshers. Worlds within worlds: Metaphors for exploring n-dimensional virtual worlds. In *Proceedings of the 3rd annual ACM SIGGRAPH symposium on user interface software and technology (UIST '90)*, pages 76–83, New York, NY, 1990. ACM Press.

[47] J.-D. Fekete and C. Plaisant. Interactive information visualization of a million items. In *2002 IEEE Symposium on Information Visualization (InfoVis 2002)*, pages 117–124, Washington, DC, October 2002. IEEE Computer Society.

[48] D. F. Ferraiolo, G.-J. Ahn, R. Chandramouli, and S. I. Gavrila. The role control center: Features and case studies. In *Proceedings of the 8th ACM Symposium on Access Control Models and Technologies (SACMAT 2003)*, pages 12–20, New York, NY, June 2003. ACM Press.

[49] K. Fishkin and M. C. Stone. Enhanced dynamic queries via movable filters. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems(CHI 1995)*, pages 415–420, New York, NY, 1995. ACM Press.

[50] K. Fisler, S. Krishnamurthi, L. A. Meyerovich, and M. C. Tschantz. Verification and change-impact analysis of access-control policies. In *Proceedings of the 27th International Conference on Software Engineering (ICSE '05)*, pages 196–205, Los Alamitos, CA, May 2005. IEEE Computer Society Press.

[51] I. Fundulaki and M. Marx. Specifying access control policies for XML documents with XPath. In *Proceedings of the 9th ACM Symposium on Access Control Models and Technologies (SACMAT 2004)*, pages 61–69, New York, NY, June 2004. ACM Press.

[52] G. W. Furnas. Generalized fisheye views. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems(CHI 1986)*, pages 16–23, New York, NY, April 1986. ACM Press.

[53] S. Godik, T. Moses, A. Anderson, B. Parducci, C. Adams, D. Flinn, G. Brose, H. Lockhart, K. Beznosov, M. Kudo, P. Humenn, S. Godik, S. Andersen, S. Crocker, and T. Moses. eXtensible Access Control Markup Language (XACML) Version 1.0. OASIS Standard, February 2003.

[54] I. Goldberg, D. Wagner, R. Thomas, and E. A. Brewer. A secure environment for untrusted helper applications: Confining the wily hacker. In *Proceedings of the 6th USENIX Security Symposium*, Berkeley, CA, July 1996. USENIX Association.

[55] N. S. Good and A. Krekelberg. Usability and privacy: a study of Kazaa P2P file-sharing. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems(CHI 2003)*, pages 137–144, New York, NY, April 2003. ACM Press.

[56] H. Hamed and E. Al-Shaer. Taxonomy of conflicts in network security policies. *IEEE Communications*, 44(3):134–141, March 2006.

[57] A. Herzog and N. Shahmehri. A usability study of security policy management. In *IFIP International Federation for Information Processing, Volume 201, Security and Privacy in Dynamic Environments: Proceedings of the IFIP TC-11 21st International Information Security Conference (SEC 2006)*, pages 296–306, Boston, 2006. Springer.

[58] M. Hochhauser. Lost in the fine print: Readability of financial privacy notices, July 2001. Available at http://www.privacyrights.org/ar/GLB-Reading.htm. Accessed on May 19, 2008.

[59] E. H. hsin Chi, P. Barry, J. Riedl, and J. Konstan. A spreadsheet approach to information visualization. In *Proceedings of the 1997 IEEE Symposium on Information Visualization (INFOVIS '97)*, pages 17–24, Washington, DC, 1997. IEEE Computer Society.

[60] R. F. Ibanez. Eiciel GNOME File ACL editor. Project website http://rofi.roger-ferrer.org/eiciel, June 2008. Accessed on June 8, 2008.

[61] T. Jaeger, R. Sailer, and X. Zhang. Resolving constraint conflicts. In *Proceedings of the 9th ACM Symposium on Access Control Models and Technologies (SACMAT 2004)*, pages 105–114, New York, NY, June 2004. ACM Press.

[62] S. Jajodia, P. Samarati, V. S. Subrahmanian, and E. Bertino. A unified framework for enforcing multiple access control policies. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, pages 474–485, New York, NY, 1997. ACM Press.

[63] S. Jayaraman and C. North. A radial focus+context visualization for multi-dimensional functions. In *IEEE Visualization 2002 (VIS2002)*, pages 443–450, Los Alamitos, CA, October 2002. IEEE Computer Society.

[64] C. Jensen and C. Potts. Privacy policies as decision-making tools: an evaluation of online privacy notices. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 471–478, New York, NY, USA, 2004. ACM.

[65] C. Jensen and C. Potts. Privacy policies as decision-making tools: an evaluation of online privacy notices. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems(CHI 2004)*, pages 471–478, New York, NY, 2004. ACM Press.

[66] B. Johnson and B. Shneiderman. Treemaps: A space-filling approach to the visualization of hierarchical information structures. In *Proceedings of the 2nd International IEEE Visualization Conference*, pages 284–291, Washington, DC, April 1991. IEEE Computer Society.

[67] A. Kapadia, G. Sampemane, and R. H. Campbell. KNOW why your access was denied: Regulating feedback for usable security. In *Proceedings of the 11th ACM Conference on Computer and Communications Security*, pages 52–61, New York, NY, 2004. ACM Press.

[68] C.-M. Karat, J. Karat, C. Brodie, and J. Feng. Evaluating interfaces for privacy policy rule authoring. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems(CHI 2006)*, pages 83–92, New York, NY, 2006. ACM Press.

[69] J. Karat, C.-M. Karat, C. Brodie, and J. Feng. Privacy in information technology: Designing to enable privacy policy management in organizations. *International Journal of Human-Computer Studies*, 63(1-2):153–174, July 2005.

[70] B. Kirwan. *A Guide to Practical Human Reliability Assessment*. Taylor & Francis, London, 1994.

[71] H. P. Kumar, C. Plaisant, and B. Shneiderman. Browsing hierarchical data with multi-level dynamic queries and pruning. In S. K. Card, J. D. Mackinlay, and B. Shneiderman, editors, *Readings in Information Visualization: Using Vision to Think*, chapter 3, pages 295–305. Morgan Kaufmann, San Francisco, CA, January 1999.

[72] J. Lamping and R. Rao. The hyperbolic browser: A focus+context technique for visualizing large hierarchies. In S. K. Card, J. D. Mackinlay, and B. Shneiderman, editors, *Readings in Information Visualization: Using Vision to Think*, chapter 4, pages 382–408. Morgan Kaufmann, San Francisco, CA, January 1999.

[73] B. W. Lampson. Protection. *Operating Systems Review*, 8(1):18–24, January 1974. Reprint of the original from *Proceedings of the Fifth Princeton Symposium on Information Sciences and Systems* (Princeton University, March, 1971), 437-443.

[74] C. E. Landwehr. Formal models for computer security. *ACM Computing Surveys*, 13(3):247–278, September 1981.

[75] S. Lederer, J. Hong, A. K. Dey, and J. Landay. Personal privacy through understanding and action: Five pitfalls for designers. *Personal and Ubiquitous Computing*, 8(6):440–454, November 2004.

[76] S. Lederer, J. I. Hong, X. Jiang, A. K. Dey, J. A. Landay, and J. Mankoff. Towards everyday privacy for ubiquitous computing. Technical Report UCB-CSD-03-1283, University of California, Berkeley, Berkeley, CA, October 2003. Available at http://www.eecs.berkeley.edu/Pubs/TechRpts/2003/CSD-03-1283.pdf.

[77] S. Lederer, J. Mankoff, and A. K. Dey. Who wants to know what when? Privacy preference determinants in ubiquitous computing. In *Extended Abstracts of the ACM SIGCHI Conference on Human Factors in Computing Systems(CHI 2003)*, pages 724–725, New York, NY, 2003. ACM Press.

[78] S. Lederer, J. Mankoff, A. K. Dey, and C. P. Beckmann. Managing personal information disclosure in ubiquitous computing environments. Technical Report UCB-CSD-03-1257, University of California, Berkeley, Berkeley, CA, 2003. Available at http://www.eecs.berkeley.edu/Pubs/TechRpts/2003/CSD-03-1257.pdf.

[79] R. Lemos. Msn sites get easy-to-read privacy label. *CNET News.com*, 2005. Available at `http://news.com.com/2100-1038_3-5611894.html`. Accessed on May 19, 2008.

[80] Y. K. Leung and M. D. Apperley. A review and taxonomy of distortion-oriented presentation techniques. *ACM Transactions on Computer-Human Interaction*, 1(2):126–160, June 1994.

[81] M. Levoy. Spreadsheets for images. In *Proceedings of the 21st annual conference on Computer Graphics and Interactive Techniques (SIGGRAPH '94)*, pages 139–146, New York, NY, 1994. ACM Press.

[82] H. R. Lipford, A. Besmer, and J. Watson. Understanding privacy settings in facebook with an audience view. In *Usability, Psychology, and Security 2008 (UPSEC 2008)*, Berkeley, CA, April 2008. USENIX Association. Available at `http://www.usenix.org/event/upsec08/tech/full_papers/lipford/lipford.pdf`. Accessed on June 22, 2008.

[83] A. C. Long, C. Moskowitz, and G. Ganger. A prototype user interface for coarse-grained desktop access control. Technical Report CMU-CS-03-200, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA, Nov. 2003. Available at http://reports-archive.adm.cs.cmu.edu/cs2003.html.

[84] M. Loy, R. Eckstein, D. Wood, J. Elliott, and B. Cole. *Java Swing*. O'Reilly, Sebastopol, CA, 2nd edition, November 2002.

[85] R. A. Maxion and R. W. Reeder. Improving user-interface dependability through mitigation of human error. *International Journal of Human-Computer Studies*, 63(1-2):25–50, July 2005.

[86] A. Mayer, A. Wool, and E. Ziskind. Fang: A firewall analysis engine. In *Proceedings of the 2000 IEEE Symposium on Security and Privacy (S&P 2000)*, pages 177–187, Los Alamitos, CA, May 2000. IEEE Computer Society Press.

[87] P. McLachlan, T. Munzner, E. Koutsofios, and S. North. LiveRAC: Interactive visual exploration of system management time-series data. In *Proceedings of the 26th annual ACM SIGCHI Conference on Human Factors in Computing Systems(CHI 2008)*, pages 1483–1492, New York, NY, April 2008. ACM Press.

[88] Microsoft Corporation. Microsoft Technet: Windows XP file permissions documentation. http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/winxppro/proddocs/acl_special_permissions.asp (current Dec. 2004).

[89] Microsoft Corporation. Best practices for permissions and user rights. Available at http://www.microsoft.com/resources/documentation/windowsserv/2003/standard/proddocs/en-us/sag_SEconceptsImpACBP.asp, 2004.

[90] Microsoft Corporation. How permissions are handled when you copy and move files and folders. Technical Report 310316, Microsoft Corporation, Redmond, WA, October 2007. Available at http://support.microsoft.com/kb/310316/en-us. Accessed on June 28, 2008.

[91] G. R. Milne and M. J. Culnan. Strategies for reducing online privacy risks: Why consumers read (or don't read) online privacy notices. *Journal of Interactive Marketing*, 18(3):15–29, Summer 2004.

[92] C. M. Mitchell and R. A. Miller. A discrete control model of operator function: A methodology for information display design. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-16(3):343–357, June 1986.

[93] R. Molich and J. Nielsen. Improving a human-computer dialogue. *Communications of the ACM*, 33(3):338–348, March 1990.

[94] M. C. Mont, R. Thyne, and P. Bramhall. Privacy enforcement with HP Select Access for regulatory compliance. Technical Report HPL-2005-10, HP Laboratories Bristol, Bristol, UK, January 2005. Available at http://www.hpl.hp.com/techreports/2005/HPL-2005-10.pdf.

[95] J. Nielsen and R. L. Mack. *Usability Inspection Methods*. John Wiley & Sons, Inc., New York, NY, 1994.

[96] D. A. Norman. *The Design of Everyday Things*. Doubleday, New York, NY, 1988.

[97] L. Palen and P. Dourish. Unpacking "privacy" for a networked world. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems(CHI 2003)*, pages 129–136, New York, NY, 2003. ACM Press.

[98] K. W. Piersol. Object-oriented spreadsheets: The analytic spreadsheet package. In *Proceedings of the 1986 Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA '86)*, pages 385–390, New York, NY, November 1986. ACM Press.

[99] S. Pocock, M. Harrison, P. Wright, and P. Johnson. Thea: A technique for human error assessment early in design. In *Proceeding of INTERACT2001 "Human-Computer Interaction - INTERACT'01"*, pages 247–254, Amsterdam, July 2001. IOS Press/IFIP. Available at http://homepages.cs.ncl.ac.uk/michael.harrison/papers/int01pub4.pdf.

[100] I. Pollach. What's wrong with online privacy policies? *Communications of the ACM*, 50(9):103–108, September 2007.

[101] R. Rao and S. K. Card. The Table Lens: merging graphical and symbolic representations in an interactive focus + context visualization for tabular information. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems(CHI 1994)*, pages 318–322, New York, NY, 1994. ACM Press.

[102] J. Reason. *Human Error*. Cambridge University Press, The Pitt Trumpington Street, Cambridge CB2 1RP, 1990.

[103] R. W. Reeder, L. Bauer, L. F. Cranor, M. K. Reiter, K. Bacon, K. How, and H. Strong. Expandable grids for visualizing and authoring computer security policies. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems(CHI 2008)*, pages 1473–1482, New York, NY, April 2008. ACM Press.

[104] R. W. Reeder, C.-M. Karat, J. Karat, and C. Brodie. Usability challenges in security and privacy policy-authoring interfaces. In *INTERACT 2007: Proceedings of the 11th IFIP TC 13 International Conference, Rio de Janeiro, Brazil, September 10-14, 2007*, Lecture Notes in Computer Science, Vol. 4663, Part II, pages 141–155, New York, NY, September 2007. Springer.

[105] Report by Kleimann Communication Group for the FTC. Evolution of a prototype financial privacy notice, 2006. Available at http://www.ftc.gov/privacy/privacyinitiatives/ftcfinalreport060228.pdf. Accessed on May 19, 2008.

[106] J. C. Roberts and M. A. E. Wright. Towards ubiquitous brushing for information visualization. In *2006 IEEE Symposium on Information Visualization (InfoVis 2006)*, pages 151–156, Washington, DC, 2006. IEEE Computer Society.

[107] G. G. Robertson and J. D. Mackinlay. Cone trees: Animated 3d visualizations of hierarchical information. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems(CHI 1991)*, pages 189–194, New York, NY, April 1991. ACM Press.

[108] J. Rode, C. Johansson, P. DiGioia, R. S. Filho, K. Nies, D. H. Nguyen, J. Ren, P. Dourish, and D. Redmiles. Seeing further: Extending visualization as a basis for usable security. In *Proceedings of the Second Symposium on Usable Privacy and Security (SOUPS 2006)*, pages 145–155, New York, NY, 2006. ACM Press.

[109] R. Rosenholtz. A simple saliency model predicts a number of motion popout phenomena. *Vision Research*, 39:3157–3163, 1999.

[110] N. Sadeh, J. Hong, L. Cranor, I. Fette, P. Kelley, M. Prabaker, and J. Rao. Understanding and capturing people's privacy policies in a mobile social networking application. *Personal & Ubiquitous Computing*, page unknown, 2008. To appear.

[111] B. Salmon, S. W. Schlosser, L. F. Cranor, and G. R. Ganger. Perspective: Semtanic data management for the home. Technical Report CMU-PDL-08-105, Carnegie Mellon University, Pittsburgh, PA, May 2008.

[112] B. Salmon, S. W. Schlosser, L. B. Mummert, and G. R. Ganger. Putting home storage management into Perspective. Technical Report CMU-PDL-06-110, Carnegie Mellon University, Pittsburgh, PA, September 2006. Available at `http://www.pdl.cmu.edu/PDL-FTP/Storage/CMU-PDL-06-110.pdf`.

[113] J. H. Saltzer and M. D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, September 1975.

[114] G. Sampemane, P. Naldurg, and R. H. Campbell. Access control for active spaces. In *Proceedings of the 18th Annual Computer Security Applications Conference*, pages 343–352, Los Alamitos, CA, December 2002. IEEE Computer Society. Available at http://www.acsac.org/2002/papers/105.pdf.

[115] R. S. Sandhu and E. J. Coyne. Role-based access control models. *IEEE Computer*, 29(2):38–47, February 1996.

[116] M. Sarkar and M. H. Brown. Graphical fisheye views of graphs. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems(CHI 1992)*, pages 83–91, New York, NY, April 1992. ACM Press.

[117] M. Sarkar, S. S. Snibbe, O. J. Tversky, and S. P. Reiss. Stretching the rubber sheet: A metaphor for viewing large layouts on small screens. In *Proceedings of the 6th annual ACM Symposium on User interface software and technology (UIST '93)*, pages 81–91, New York, NY, 1993. ACM Press.

[118] M. Scaife and Y. Rogers. External cognition: how do graphical representations work? *International Journal of Human-Computer Studies*, 45(2):135–262, August 1996.

[119] J. W. Senders and N. P. Moray. *Human Error: Cause, Prediction, and Reduction*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1991.

[120] H. Shen and P. Dewan. Access control for collaborative environments. In *Proceedings of the 1992 ACM conference on Computer-supported cooperative work (CSCW)*, pages 51–58, New York, NY, October 1992. ACM Press.

[121] B. Shneiderman. Direct manipulation: A step beyond programming languages. *Computer*, 16(8):57–69, August 1983.

[122] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the 1996 IEEE Workshop on Visual Languages*, pages 336–343, Los Alamitos, CA, 1996. IEEE Computer Society.

[123] B. Shneiderman, D. Feldman, and A. Rose. Visualizing digital library search results with categorical and hierarchical axes. Technical Report UMIACS-TR-99-12, University of Maryland, College Park, MD, February 1999. Available at http://hcil.cs.umd.edu/trs/99-03/99-03.pdf.

[124] R. Smith. Personal Communication, March 2004.

[125] M. Spenke, C. Beilken, and T. Berlage. FOCUS: the interactive table for product comparison and selection. In *Proceedings of the 9th annual ACM Symposium on User interface software and technology (UIST '96)*, pages 41–50, New York, NY, 1996. ACM Press.

[126] M. Stiegler, A. H. Karp, K.-P. Yee, and M. Miller. Polaris: Virus safe computing for Windows XP. Technical Report HPL-2004-221, HP Laboratories, Palo Alto, CA, December 2004. Available at http://www.hpl.hp.com/techreports/2004/HPL-2004-221.pdf. Accessed on June 18, 2008.

[127] J. Stoll, C. S. Tashman, W. K. Edwards, and K. Spafford. Sesame: Informing user security decisions with system visualization. In *Proceedings of the 26th annual ACM SIGCHI Conference on Human Factors in Computing Systems(CHI 2008)*, pages 1045–1054, New York, NY, April 2008. ACM Press.

[128] L. Takayama and E. Kandogan. Trust as an underlying factor of system administrator interface choice. In *Conference on Human Factors in Computing Systems(CHI 2006) Extended Abstracts*, pages 1391–1396, New York, NY, April 2006. ACM Press.

[129] The Open Group Research Institute. Adage system overview. Available at http://www.memesoft.com/adage/SystemSpec.ps. Accessed via HTTP on September 20, 2006.

[130] J. Theeuwes. Perceptual selectivity for color and form. *Perception and Psychophysics*, 51(6):599–606, 1992.

[131] J. Tsai, S. Egelman, L. Cranor, and A. Acquisti. The effect of online privacy information on purchasing behavior: An experimental study. In *The 6th Workshop on the Economics of Information Security (WEIS)*, 2008. Available at http://weis2007.econinfosec.org/papers/57.pdf. Accessed on February 26, 2008.

[132] U.S. Senate Sergeant at Arms. Report on the investigation into improper access to the Senate Judiciary CommitteeŠs computer system. Available at http://judiciary.senate.gov/testimony.cfm?id=1085&wit_id=2514, March 2004.

[133] J. J. van Wijk and R. van Liere. Hyperslice: Visualization of scalar functions of many variables. In *IEEE Visualization 1993 (VIS1993)*, pages 119–125, Los Alamitos, CA, 1993. IEEE Computer Society.

[134] A. Varshney and A. Kaufman. FINESSE: a financial information spreadsheet. In *Proceedings of the 1996 IEEE Symposium on Information Visualization (INFOVIS '96)*, pages 70–71, Washington, DC, 1996. IEEE Computer Society.

[135] C. Ware. *Information Visualization: Perception for Design*. Morgan Kaufman, San Francisco, CA, 2000.

[136] C. Ware. *Information Visualization: Perception for Design*, chapter Visual Attention and Information that Pops Out, pages 151–199. Morgan Kaufman, San Francisco, CA, 2000.

[137] L. Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer, New York, NY, 2004.

[138] T. Whalen, D. Smetters, and E. F. Churchill. User experiences with sharing and access control. In *Conference on Human Factors in Computing Systems(CHI 2006) Extended Abstracts*, pages 1517–1522, New York, NY, April 2006. ACM Press.

[139] A. Whitten and J. Tygar. Why Johnny can't encrypt: A usability evaluation of PGP 5.0. *Proceedings of the 8th USENIX Security Symposium*, page unknown, August 1999. August 23-26, 1999, Washington, DC.

[140] A. Whitten and J. Tygar. Safe staging for computer security. Presented at Workshop on Human-Computer Interaction and Security Systems, part of ACM SIGCHI Conference on Human Factors in Computing Systems (CHI 2003), April 2003. Available at `http://www.andrewpatrick.ca/CHI2003/HCISEC/hcisec-workshop-whitten.pdf`.

[141] D. Wiegmann and S. Shappell. *A Human Error Approach to Aviation Accident Analysis*. Ashgate Publishing Co., Aldershot/Burlington, 2003.

[142] G. J. Wills. Selection: 524,288 ways to say "this is interesting". In *1996 IEEE Symposium on Information Visualization (InfoVis 1996)*, pages 54–60, Washington, DC, 1996. IEEE Computer Society.

[143] D. D. Woods. Paradigms for intelligent decision support. In E. Hollnagel, G. Mancini, and D. D. Woods, editors, *Intelligent Decision Support in Process Environments*, pages 153–173. Springer-Verlag, Berlin, 1985.

[144] D. D. Woods and E. M. Roth. Cognitive systems engineering. In M. Helander, editor, *Handbook of Human-Computer Interaction*, chapter 1, pages 3–43. Elsevier Science Publishers B.V., Amsterdam, The Netherlands, 1st edition, 1988.

[145] A. Wool. A quantitative study of firewall configuration errors. *IEEE Computer*, 37(6):62–67, June 2004.

[146] K.-P. Yee. User interaction design for secure systems. In *Information and Communications Security, 4th International Conference, ICICS 2002, Singapore*, Lecture Notes in Computer Science, Vol. 2513, pages 278–290, New York, NY, December 2002. Springer. Available at http://www.sims.berkeley.edu/ ping/sid/uidss-icics.pdf.

[147] K.-P. Yee. Aligning security and usability. *IEEE Security and Privacy*, 2(5):48–55, September 2004.

[148] L. Yuan, J. Mai, Z. Su, H. Chen, C.-N. Chuah, and P. Mohapatra. FIREMAN: A toolkit for FIREwall Modeling and ANalysis. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy (S&P 2006)*, pages 199–213, Los Alamitos, CA, 2006. IEEE Computer Society Press.

[149] J. Zhang. A representational analysis of relational information. *International Journal of Human-Computer Studies*, 45:59–74, 1996.

[150] J. Zhang. The nature of external representations in problem solving. *Cognitive Science*, 21(2):179–217, 1997.

[151] J. Zhang and D. A. Norman. Representations in distributed cognitive tasks. *Cognitive Science*, 18(1):87–122, 1994.

[152] M. E. Zurko. Adage usability testing results: Formal testing affinity mapping and questionnaire. Available at http://www.memesoft.com/adage/affinity.ps. Accessed via HTTP on September 20, 2006.

[153] M. E. Zurko, R. Simon, and T. Sanfilippo. A user-centered, modular authorization service built on an RBAC foundation. In *Proceedings 1999 IEEE Symposium on Security and Privacy*, pages 57–71, Los Alamitos, CA, May 1999. IEEE Computer Security Press.

[154] M. E. Zurko and R. T. Simon. User-centered security. In *Proceedings of Workshop on New Security Paradigms*, pages 27–33, Lake Arrowhead, CA, September 1996. Available at http://www.memesoft.com/adage/.

About the Author

Rob Reeder started his PhD in the Computer Science Department at Carnegie Mellon in the fall of 2001 and has worked with his advisor, Lorrie Cranor, in the CMU Usable Privacy and Security (CUPS) Lab. He currently does research at the intersection of human-computer interaction and security and privacy, but he maintains broad research interests in all aspects of HCI, information visualization, security, and privacy. Prior to starting graduate school, Rob worked in the User Interface Research group at Xerox PARC on information visualization and Web usability and was a competitive marathoner at the national level. He won the 2000 Las Vegas Marathon and competed in the 2000 U.S. Olympic Trials marathon. Rob received a B.S. and M.S. in Computer Science from Stanford University in 1997 and 1999, respectively. At Stanford, he was captain of the 1996 cross country team that won the NCAA Division I national championship, and he was a track and field All-American at the 5K and 10K.