RBAC what? Development of a role-based access control policy writing tool for e-Scientists.

Sacha Brostoff, M. Angela Sasse[a] and
David Chadwick, James Cunningham, Uche Mbanaso, O. Otenko[b]

[a] Department of Computer Science, UCL (University College London)

[b] Information Systems Institute, University of Salford

Abstract

An access control policy writing tool for the PERMIS role-based privileges management infrastructure was iteratively developed employing usability principles and techniques. Expert and intermediate users' efficiency in policy creation was improved. Three novice users took part in a usability trial with the first prototype, attempting to recreate a simple policy in 15 minutes that had been specified in plain English. The participants had not properly understood the labelling of buttons or fields in the interface, and so experienced difficulty in breaking down the policy into components and identifying parts of the application to put them in. The non-specialists found it challenging to express access policy effectively because their concept of it did not match what was presented to them on screen. Bubble help and alert boxes were expanded and made more prescriptive to guide their actions without impacting expert users' efficiency. *Conceptual design* techniques were used to revise the labels based on potential users' descriptions of RBAC. A questionnaire study showed improved label intuitiveness (t=6.28, df=7, p=.000 two tailed): e-Scientists and developers were better able to describe access policy components from labels, and match labels with components. This project has successfully developed an access control tool to improve security specialists' productivity and improve the wider e-Science community's access to a flexible security infrastructure.

1 Introduction

This paper tracks predicted user experiences and performance of an e-Science security software package over the course of its multistage development. Usability depends on the context in which it is measured and has several indices, including *efficiency* (time to write policy, use of help facilities, etc.), *effectiveness* (e.g. task completion, quality of resulting policy, etc.), and *user satisfaction* (BSI, 1998). This paper concentrates on *learnability* - reducing the initial learning barrier that users must surmount to become productive - an aspect of usability that crosses the three measures above.

Stages of our authorisation policy writing software's development focusing on usability issues are outlined, along with each stage's methods, their results and subsequent diagnoses for improvements. We close with the distillation of our positive and negative experiences of introducing usability into the design process.

1.1 What is PERMIS

PERMIS (Chadwick, Otenko, & Ball, 2003) is an infrastructure that allows role-based access control to be implemented in e-Science systems, such as but not limited to computational grids. PERMIS consists of an application gateway surrounding the grid node which looks up users' x.509 attribute-certificates (ACs) in an LDAP directory in response to their requests for access. The attribute certificates contain the users' roles, which are compared against a policy on the grid node listing the permitted roles, and a decision to grant or deny access is made. A check is also carried out that the entity that issued the AC~~certificate~~ to the user (with the PERMIS Privilege Allocator tool) was authorised to do so.

PERMIS greatly simplifies access control for managers of grid nodes through the use of the role based access control paradigm - privileges are given to roles rather than individual users. Whilst the individuals who fulfil roles in an organisation may come and go with relative frequency, the roles themselves are relatively unchanging. In this paradigm, the access control task is simplified to giving privileges to roles, with an additional job of distributing roles to users. The management task is further reduced by delegating~~ion of~~ the task of distribution of ACs ~~job~~ to named individuals. ~~(~~called Sources of Authority (SOAs). The manager of a grid node need only write the names of the SOAs into the access control policy at his/her node. ~~SOAs (managers)~~ and then (s)he can leave it up to these SOAs (managers) in other locations or institutions comprising the grid project to~~can~~ distribute the ACs to their own users.

1.2 Why the policy editor

PERMIS access control policies are currently written by hand in XML. XML policies have several advantages:

- they are machine processable and can be easily parsed and understood by the access control engine;
- the content of them can be controlled by a DTD or schema thus ensuring that the manager creates syntactically correct policies;
- open source tools are available for manipulating XML data structures thereby reducing implementation costs;
- XML policies can be automatically generated by applications (for example in an open tendering application developed in the PERMIS project, the tender application generated a specific policy governing the submission proves for each tender).

~~{Salford to complete please!}~~

Formatted: Bullets and Numbering

However, PERMIS developers considered that writing XML would not be *efficient* for grid managers, for the following reasons:

1. We predicted that the users would be e-Scientists-unfamiliar with RBAC, PERMIS and XML.
2. The opportunity for error - XML requires a strict grammar be followed which users may make mistakes in;
3. Recall rather than recognition – users would have to remember the correct tags, rather than recognise them;

~~4.~~

~~5.~~4. Effort - machine readable security policies require interpretive infrastructure (such as tags to denote different classes of policy items) to avoid ambiguity, which tend to add extra length to the resulting policy. PERMIS policies are brief compared with alternatives such as Akenti's or XACML (OASIS 2002), but still require two pages of

XML for a policy that can be written in a few sentences of English.

In addition, since the intended user's were predicted to know nothing about RBAC PERMIS,

6.5. Learnability- a graphical user interface (GUI) should make PERMIS and RBAC concepts easier to learn so users can be productive more quickly.

It was therefore decided to build a tool to produce the XML for grid managers, assisting them with RBAC concepts and leaving them to concentrate on the semantics of the policy. Furthermore, it was decided to use a graphical user interface, so that this interaction style's best features could be used to full effect, for example by using drag-and-drop and point-and-click to reduce the effort used in policy construction.

2 First prototype

2.1 Design features

The policy editor toolbar is the main focus for interaction (Figure 1). Users click on buttons in the toolbar to open windows - forms that are used to fill in parts of the access control policy - and the names for these have been taken directly from the technical language of PERMIS policy elements. The *Subject* policy window is shown in Figure 1.

Each grid protected by PERMIS will refer to or contain at least one LDAP directory[1], which is a repository for the x.509 attribute certificates used to store users' roles. The policy editor is configured to browse these LDAP directories, and represent them as trees in its domain windows (subject , SOA, and target buttons) - see lower left-hand corner of Figure 1.

In normal operation the user browses the directory tree until finding the desired entry (for example University College London Department of Computer Science), which can be double clicked to enter it into the *LDAP DN* field. The user indicates that this DN should be included in the subject domain policy by clicking the *include* button. The policy writer then types a nickname for the subject domain he is defining in the *Domain ID* field, for example "UCL", which he will have to refer to in other windows. The information is added to the policy by clicking the *OK* button, and the window is then closed by clicking the *Close* button.

In some cases the LDAP directories will not be available for some reason[2], so no LDAP tree will appear in the window. The user will then have to type in DNs into the *LDAP DN* field from memory instead of seeing, pointing and clicking.

The two most important policy windows are the *Target Access* policy window and the *Role Assignment* window (Figure 2). These windows refer to policy items defined in the other windows by using combo-boxes (see *SOA Roles,* and *Subject* boxes in Figure 2). The combo-boxes contain the ID nicknames users created in previous windows - so for example if the user typed UCL into the domain ID field in Figure 1, it could be selected in the target domain combo-box in Figure 2, and that the user was referring to the subject domain he defined as University College London's Computer Science Department.

[1] the grid may use an external LDAP directory – e.g. LDAP directory of a collaborating institution. In extreme cases a grid may not need its own LDAP server, but usually it does.

[2] Some of the reasons are: a) security restrictions – LDAP directory may not be publicly browsable, b) firewalls - attempting to connect from a location disallowed by the institutional policy, c) network unavailability – the manager is working on the policy whilst on a plane or train

2.2 Efficiency with intermediate to expert users

Informal comparisons by UCL and PERMIS developers showed that their first prototype dramatically improved the efficiency with which PERMIS policies could be written. At least one developer reported that policy creation times were shortened by more than 530%. The prototype therefore showed the planned improvement in usability for users who are familiar both with it and PERMIS policies - predicting a considerable increase in productivity for intermediate and expert users.

It was therefore decided to focus further development efforts on improving *learnability*, to increase the speed with which novice users could progress to intermediate status, and achieve higher levels of satisfaction with the tool earlier.

2.3 Usability trial study

Three research fellows from UCL were recruited who did not know anything about PERMIS, nor who had specialist knowledge of computer security (a biological anthropologist who uses computationally intensive statistical analyses, a Human-Computer Interaction researcher, and a grid developer). The policy editor was installed on a workstation in an open plan office, with screen recording software and a microphone to record audio.

Participants went through the trial one at a time. The test facilitator described the procedure to the research fellows (participants), mentioning the recording, and consent was sought to continue. Participants were then briefed that they were about to use software for writing access control policies, but not given further information about the architecture of PERMIS, nor given information about role-based access control. They were then given a scenario sheet containing a simple access control policy written in natural language (see Table 3), and given 15 minutes to implement this using the policy editor.

The test facilitator sat by the side of each participant writing down observations and asking them to think aloud, and prompting them to describe their thoughts whenever they appeared confused or attempted incorrect actions. The participants were not given access to a manual or help materials (none existed at this point, nor were they created for the trial). The test facilitator was evasive when asked for help, answering in generalities rather than specifics, and asking participants to continue for a few more minutes where appropriate. At the end of the allotted time, the facilitator stopped the participants, and gave them a debriefing interview.

The lead developer objected in principle to this methodology, arguing that it used the wrong set of participants, or at least, did not give them sufficient background knowledge before they started. Because they did not understand the structure of LDAP DNs or understand the vocabulary used in policy based authorisation – it was argued that they were set up to fail, and implied that computer managers should have been used instead – this further implied that computer managers would have the necessary background knowledge. It was proposed that this background knowledge be measured if possible. In contrast, the test designer believed that users of the PERMIS policy editor would have the title of computer manager thrust upon them, rather than attain it through expertise, implying that the PERMIS system (including graphical user interface, help and training materials) should supply the knowledge necessary to create policies. This issue remains open and cannot be resolved until the policy editor is released and its users questioned about their background.

A further objection was raised, that 20 minutes was unrealistically short for each trial's total duration. In real use someone writing policies with the software would not have such short deadlines, and would be able to spend more time to learning the user interface. These are powerful criticisms. The test designer argued that these flaws were unavoidable due to pragmatics – recruitment of volunteers would be extremely difficult for trials of realistic duration, and existing users could not be observed in an ethnographic study as none yet exist. It was further argued that the trials would still reveal useful information, particularly relating to

the initial barriers to use of the policy editor, though some longer term usability issues would also be quickly observable.

The test scenario could be completed by the test facilitator in five minutes. None of the participants completed the scenario after 15 minutes. The grid developer was given extra time, and had not finished the scenario after 30 minutes when the trial was stopped.

There were four main reasons that participants found it difficult to complete the task. Three of these were related to terminology, and the fourth was a difficulty in writing well-formed LDAP distinguished names. These difficulties are described below.

Toolbar and window labels. None of the participants understood the label *SOA*. Two of the participants did not understand *RBAC*. One of the participants thought that what should have been entered into the *SOA* window was a role that should in fact be defined in the *Role Hierarchy* policy. One of the participants correctly identified the purpose of the *Subject* and *Targets* policies, but ascribed the purpose of the subject policy to the target policy, and vice versa.

ID field labels. All three participants believed that the ID fields (in the *target*, *subject*, *SOA*, *target access* and *role assignment* windows) had some pre-existing correct value that they didn't know, and therefore could not input. These fields were intended to collect a value which users would find easy to recognise when seen in other windows. One of the three, who had guessed the purpose of the *RBAC* policy ID field, incorrectly copied the policy's object ID number into all of the other ID fields in the application.

Exclude button. The scenario required that people from Germany be prevented from using a resource, which should have been achieved by not specifying Germany as a subject domain (PERMIS denies all access except that which has been explicitly allowed). The only participant who attempted this part of the scenario incorrectly tried to define a subject domain with Germany as an excluded node. This mistake showed that the user did not fully understand the function of the exclude button (see Table 2), since exclude can only be used after a superior node has been included. Nor did he understand the underlying *deny all access except* nature of PERMIS policies[3]. This latter misunderstanding may be a common issue for people who are not intimately familiar with access control lists or similar mechanisms which only allow people to have access (rather than allow and deny access). ~~However, the technical terminology appeared to support the misunderstanding, rather than correct it.~~The role that labelling plays in understanding this common property of access control policies should be further studied, as it may be key in making the misunderstanding less common.

LDAP distinguished names. None of the participants were able to write syntactically correct distinguished names (DNs) to refer to authorisation *targets*, *subject* domains or *sources of authority*. Whereas the previous problems would be serious in all predicted uses of the policy editor, the LDAP DN syntax problem would only occur in the unusual circumstance that the policy editor would not able to browse the grid's LDAP directory. However, this result is a cause of concern should this circumstance arise.

3 The second prototype

3.1 Design features

The second prototype retains the overall look and feel of its predecessor. However, an effort was made to make the language in the interface more intuitive for people with less background

[3] The observed use of the Exclude button on Germany would have been correct if a superior node had first been *included*. However, a superior node was available ("Directory" –see Figure 1) and the user did not seek to *include* it nor create an alternative superior node such as "the world" to *include*.

in role-based access control and PERMIS, and to make it more prescriptive:

1. Many of the labels and bubble help text were changed, and where the labels were not changed the bubble help was (see Figure 3 and Table 2).
2. The alert boxes were also changed to make them more intuitive (see Table 4).
3. Steps were taken to improve users' understanding of the nature of PERMIS policies, so that confusion about the function of the exclude button could be avoided (Figure 3).

~~4.A new field was added to the Target (now Resource) domain window (Figure 4).~~

Formatted: Bullets and Numbering

The first prototype only contained a subset of the functionality of the PERMIS policy, and an additional function - the ability to select between types of resources in a domain - was added to the Target (now Resource) domain window (see Figure 4). In addition, t~~T~~he new labels, bubble helps and alerts were created by using the Java resource bundle class. This allows the display strings to be written in a table, with keys identifying each row of the table. Column are used to hold the display strings in different languages (locales), so that the interface becomes multilingual and can be easily switched to another language by simply changing the locale.

3.2 Analytic study

The metaphor evaluation matrix (cf. Anderson et al., 1994) was applied to the first and second prototype's labels by one team member. This qualitative technique requires the analyst to fill cells in the matrix for a label or metaphor through a process of guided introspection, taking the user's point of view (Clark & Sasse, 1997). Empirical techniques (interviews, questionnaires, etc.) can be used to supplement introspection if required.

The second prototype's labels such as *Users, Managers, Resources,* and *Access Control* gave a large improvement in the amount of supported functionality that they implied (i.e. M+S+ cell entries). For example, prototype one's *Subject* label would imply specification of something to which the policy applied, but would not be enough for a non-specialist user to understand what kind of thing should be specified, whereas its prototype two version *Users* would imply the specification of persons who wish to use the protected resource - almost a perfect match for this part of the system's functionality.

However, the analysis suggested that prototype two labels contained *conceptual baggage* that would cause misunderstandings with some users. For example: *Users* implies the specification of individual persons who wish to use a protected resource - something which cannot be specified in PERMIS policies, because PERMIS gives access to roles rather than individuals.

Table 1. A metaphor evaluation matrix (cf. Anderson et al., 1994)

	M+	M-
S+	*Desirable* Features provided by the system and supported by the metaphor. Leads to correct use of system.	*Undesirable* Features provided by the system and not supported by the metaphor. Leads to underused features.
S-	*Very undesirable* Features implied by the metaphor but not supported by the system: *Conceptual baggage*	*Not important* Features not implied by the metaphor and not supported by the system

	Leads to user errors.	

4 The third prototype

4.1 Design features

The eight main button labels were revised using the metaphor evaluation matrix, and minor bug fixes made. The revised labels can be seen in Table 2 and Table 5.

4.2 Questionnaire Study

The questionnaire was designed to test the intuitiveness of the version 2 and 3 labels. Each questionnaire was in two sections. The first section gave the labels and a policy goal, and asked respondents to describe and give an example of what should go in the relevant window. This gave a sense of respondents understanding of PERMIS functionality from knowing the labels alone. Tthe second section of the questionnaire gave each label, a different policy goal, and policy statements that had been derived from the goal - one per label. Respondents were asked to match labels to policy statements by drawing a line between them. This measurement is similar to the first one but gives respondents extra cues that are like those that would be available when looking at the actual prototype.

A call to participate was sent out by email to UCL's and the University of Cambridge's internal mailing lists for people who were interested in e-Science (predominantly e-Scientists, system administrators and e-Science researchers/developers)– approximately 207 people. In addition, a call to participate was sent out to the TrustCoM project (80 people), and 3 people who had downloaded PERMIS and had previously answered a requirements questionnaire. No material inducements were given to participate (entry into a prize draw, etc.). 8 responses were received, giving a response rate of 2.8%.

The scores for each Section were aggregated. Version 2 labels scored an average of 9.25 out of 16 (57.8%; minimum score = 6.5, maximum =10.5), and version 3 labels scored an average of 13.9 out of 16 (86.7%; minimum score =12, maximum =16). This difference was statistically significant (paired t=6.28, df=7, p=.000 two tailed), showing that version 3 labels are more intuitive than version 2 labels, with a large effect size (f=0.78).

The improvements in intuitiveness scores of individual labels varied greatly between versions two and three (Table 5). *User privileges*, *Account administrator privileges* and *Resource functionality* showed particularly good improvements. The relatively low score for *resource functionality* in particular was a matter of concern, leading to a reassessment of the entire set and production of a 4th set of labels (see appendix) to be tested in an upcoming usability trial.

5 Discussion

The developers set out to improve the usability of the PERMIS system for writing access control policies, and by so doing made PERMIS more accessible to the community at large. They have achieved this goal using four techniques:

Allocation of function - the task of writing XML was reallocated from the user to software, thus producing a large increase in the efficiency with which intermediate and expert users could create policies.

Usability trials - identified particular features of the user interface that could be altered to improve the speed which novice users could appreciate the functions of different parts of the software.

Conceptual design - generated design ideas and informed design decisions for improving learnability for novice users, and so lowering the costs of adopting our software.

Low fidelity prototyping/questionnaire studies - were used to test design ideas without using precious programming resources.

We've noted a number of points during this development that other e-Science security projects may wish to consider. Incorporating established usability techniques improved our development process and our resulting software (see above), and having usability specialists on the team made this easier. In particular, we found that testing often and early was of benefit, as it gave us powerful insights into design when we had resources to do something about them. The availability of usability data from testing shortened arguments about design decisions. In contrast, design discussions could become quite drawn-out where data was not available. This was particularly an issue with the analytic (rather than empirical) usability technique *heuristic evaluation* (Nielsen, 1994a, 1994b), which we later found to have made valid predictions.

We were surprised by the difficulty of recruiting test participants from among the e-Science community - our calls for participation had a 2.8% response rate for questionnaire studies, and a 0% response rate for usability trials. User trials could only be conducted by using face-to-face meetings for recruiting. As a result, we were not able to achieve the level of testing we desired, and this made our development process more difficult. PERMIS and e-Science are relatively new - there were few members of an existing user group that would be motivated by self-interest (in having improved software) to take part in testing - participation in usability trials can take up to an hour. We were forced to create low impact test instruments to make participation more attractive - short questionnaires that implemented some aspects of low fidelity prototype testing. However, we found these to be an excellent way of collecting data - quick, cheap, informative, and requiring no programming resource. Having other means of attracting test participants would have been beneficial - budgeting to pay for test participants, or having a large group of project partners who could be put under social or contractual pressure to participate. Future projects may make more extensive use of usability inspection methods - such as *heuristic evaluation* and *cognitive walk-through* or its more streamlined derivatives (e.g. Spencer, 2000) - which do not require any test participants, but can identify significant numbers of issues revealed by usability trials (Desurvire, 1994).

Whilst some usability interventions (such as *reallocation of function* of XML creation from the user to the computer) required deep changes to our software - other improvements were almost cosmetic in terms of the coding required to implement them (changes to button labels lead by *conceptual design*) - but nevertheless brought significant improvements to usability. Good software engineering practices made these improvements easier to implement - however, such practices have much wider benefits, for example making software localisation more efficient.

The utility of our questionnaire studies showed that it was not necessary to have working code to get useful design data. We conclude that in some instances it can be preferable to test in the absence of a working prototype, as design ideas can be tested without throwing code away (Snyder, 2003). This leads to the conclusion that testing and freezing of user interface designs should be done as much as possible *before* coding begins, or if not then as early as possible - giving the developers greater flexibility in design, and a more efficient and often more rewarding software engineering process (Cooper, 1999).

There is a perception that the difficulty in getting e-Science technology to work drives security considerations to the back of the queue, and that security mechanisms impede application users rather than support them. This perception causes users to find ways of subverting the security mechanism (Sasse, Brostoff, & Weirich, 2001). We feel that

incorporating usability principles and techniques into the development of e-Science security software can make it more attractive to the wider e-Science community, increasing its uptake and appropriate use.

References

Anderson, B., Smyth, M., Knott, R. P., Bergan, M., Bergan, J., & Alty, J. L. (1994). *Minimising conceptual baggage: making choices about metaphor.* Paper presented at the People and computers IX, Glasgow.

BSI. (1998). *Ergonomic requirements for office work with visual display terminals (VDTs). Guidance on usability* (BS EN ISO 9241-11:1998). London: BSI.

Chadwick, D. W., Otenko, A., & Ball, E. (2003). Implementing Role Based Access Controls Using X.509 Attribute Certificates. *IEEE Internet Computing*, 62-69.

Clark, L., & Sasse, M. A. (1997, August). *Conceptual Design Reconsidered - The Case of the Internet Session Directory Tool.* Paper presented at the HCI '97, Bristol.

Cooper, A. (1999). *The Inmates Are Running the Asylum: Why High-tech Products Drive Us Crazy and How to Restore the Sanity*: Sams.

Desurvire, H. W. (1994). Faster, cheaper!! Are usability inspection methods as effective as empirical testing? In J. Nielsen & R. L. Mack (Eds.), *Usability inspection methods* (pp. 173-202). New York: John Wiley & Sons, Inc.

Nielsen, J. (1994a). Heuristic Evaluation. In J. Nielsen & R. L. Mack (Eds.), *Usability inspection methods* (pp. 25-62). New York: John Wiley & Sons.

Nielsen, J. (1994b). *How to Conduct a Heuristic Evaluation*, [Web page]. Available: http://www.useit.com/papers/heuristic/heuristic_evaluation.html [2003, 9th December].

OASIS (2002) *OASIS eXtensible Access Control Markup Language (XACML) v1.0*, 12 Dec 2002, available from http://www.oasis-open.org/committees/xacml/

Sasse, A., Brostoff, S., & Weirich, D. (2001). Transforming the 'weakest link' — a human-computer interaction approach to usable and effective security. *BT technology journal, 19*(3), 122-131.

Snyder, C. (2003). *Paper prototyping: The fast and easy way to design and refine user interfaces*. London: Morgan Kaufmann.

Spencer, R. (2000, April 1-6). *The Streamlined Cognitive Walkthrough Method, Working Around Social Constraints Encountered in a Software Development Company.* Paper presented at the CHI 2000, The Hague, The Netherlands.

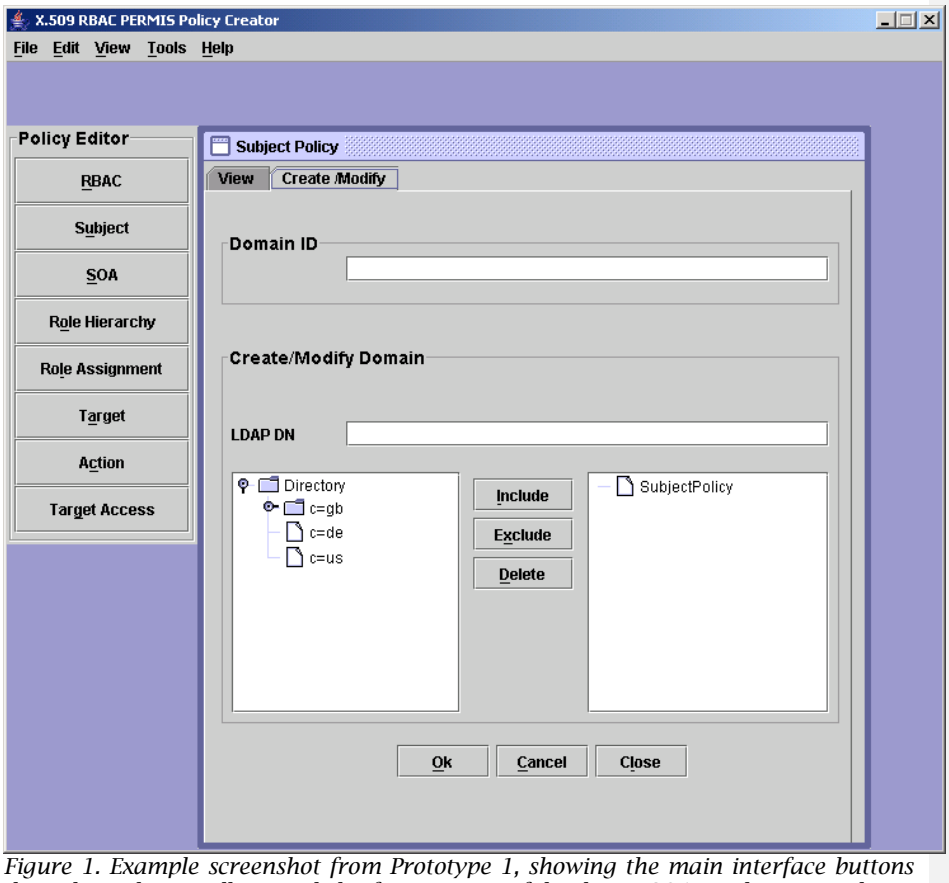

Figure 1. Example screenshot from Prototype 1, showing the main interface buttons in the policy editor toolbar, and the form type useful subject, SOA, and target policies. LDAP directories can be expanded and collapsed (lower left-hand corner of window).

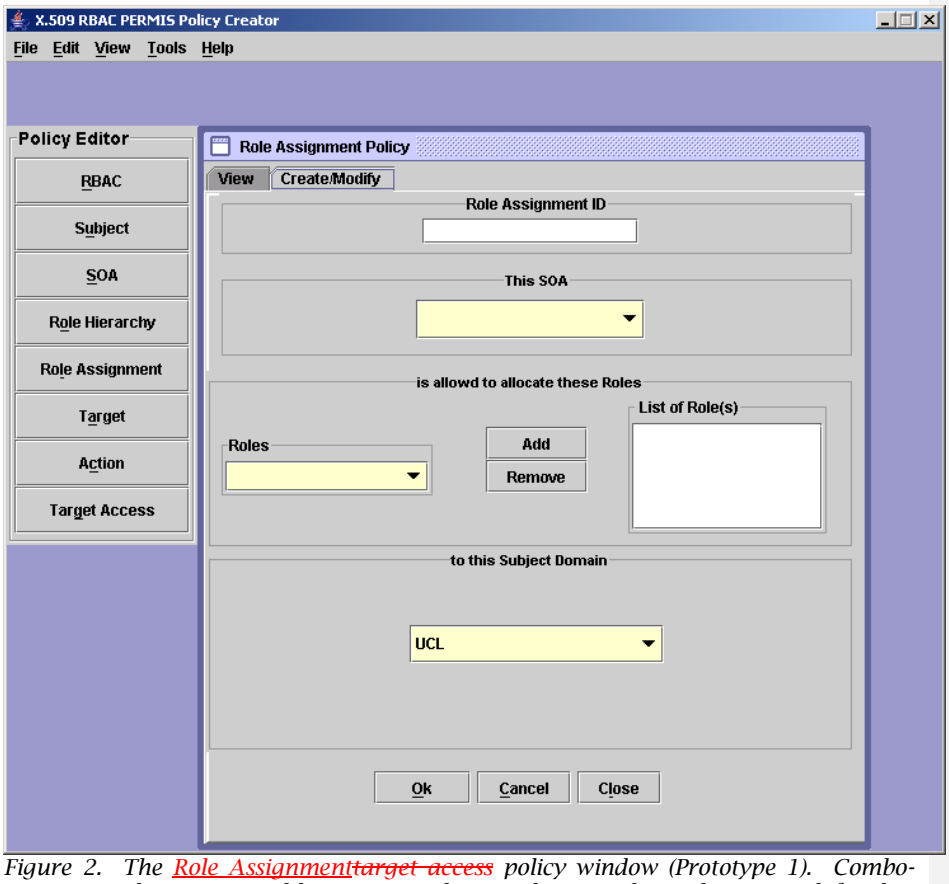

Figure 2. The Role Assignment ~~target access~~ policy window (Prototype 1). Combo-boxes contain the IDs created by users in other windows, so that policy items defined in these other windows can be referred to.

Table 2.labels and bubble help for prototypes versions 1 and 2~~two~~, with a description of each interface items purpose. Keyboard shortcuts have been removed from bubble help columns as they have not been changed. Revised bubble help for v3 and v4 not shown as changes were not that significant

Labels v1	Bubble help v1	Labels v2	Bubble help v2	Labels v3	Labels v4	Meaning/Purpose
Rbac	Edit RBAC policy	Policy Number	Select this to enter the unique policy number	Policy object ID	*No change*	Enter the unique Policy OID
Subject	Edit Subject Policy	Users	Select this to define or view who the groups of users are	Where users are from	*No change*	~~A branch of~~ ~~a~~An LDAP directory domain~~tree~~ in which users reside (a subject domain). This will ~~which is~~subsequently be put under the authority of an SOA to ~~upload~~ allocate roles (Attribute Certificates) to.
SOA	Edit SOA Policy	Managers	Select this to define or view who the trusted managers are	Account administrators	*No change*	"Source of Authority" – a person you delegate ~~to~~ the task of assigning role ACs to users.
Role hierarchy	Edit Role hierarchy Policy	Roles	Select this to define or view the roles covered by this policy	User types	User roles	Naming of roles and the d~~D~~efinition of the role hierarchy (i.e. inheritance-of-privileges relationships between roles)~~them~~
Role assignment	Edit Role assignment Policy	Role allocations	Select this to define or view which trusted managers can assign which roles to which group of users	Administrator privileges	*No change*	Which roles an SOA can assign to which subject domains ~~for which targets~~
Target	Edit Target Policy	Resources	Select this to define or view what resources are to be p~~r~~rotected by this policy	My protected resources	*No change*	An ~~branch or leaves of an~~ LDAP directory domain~~tree~~ which contains Grid nodes owned by the grid node manager that are to be protected by the policy (target domain)

Labels v1	Bubble help v1	Labels v2	Bubble help v2	Labels v3	Labels v4	Meaning/Purpose
Action	Edit Action Policy	Actions	Select this to define or view the actions that users are allowed to perform on the resources	Resource functionality	Resource~~'~~s' functions	Things that users can request to do on targets, about which an access control decision must be made
Target access	Edit Target access Policy	Access control	Select this to define or view which actions can be performed on which resources by which roles	User privileges	*No change*	Which roles can do which actions on which targets
Include	Include in the subject domain spec	Include	Select this to include the domain entered in the box above in the SubjectPolicy	*No change*	*No change*	Indicate that you wish an LDAP subtree branch to become part of a subject or target or SOA domain~~s~~.
Exclude	Exclude from the subject domain spec	Exclude	Select this to exclude the domain entered in the box above from the selected Include domain	*No change*	*No change*	Indicate that you wish to snip out a sub branch from an LDAP tree branch that you have included.
Delete	Delete this node	Delete	Select this to delete the item selected in the right hand window	*No change*	*No change*	Delete branch or leaf from Subject Policy box, SOAPolicy box, or TargetPolicy box.
OK	Press button to put contents into XML tree	OK	Select this to add the data from the form to the policy in memory	*No change*	*No change*	Add the form contents to the access control policy, keeping the form/window open.
Cancel	Cancel without putting content to tree	Cancel	Select this to remove the data from the form without adding anything to the policy in memory	*No change*	*No change*	Clears all the fields in the form.
Close	Close this window	Close	Select this to close the form	*No change*	*No change*	Closes the currently open form. Does not check if form contents have been saved.

Labels v1	Bubble help v1	Labels v2	Bubble help v2	Labels v3	Labels v4	Meaning/Purpose
Remove	Remove from list of Actions domain	Remove	Remove the selected action from the window above	*No change*	*No change*	~~Clears the~~ Remove the Action ~~fields in~~from the list in the form
Add		Add	Select this to ~~define or view~~add …. (contents varied depending upon context)	*No change*	*No change*	~~Move or copy text from one field to another, without adding it to the access control policy~~Add the selected item to the appropriate list (dependent upon context)

Table 3. Scenario for User trials of prototype 1

"UCL's "Computer Science" has a really great networked printer, called "printer". The head of department has put you in charge of access control for it. He's instructed you to write a policy for it that will delegate to "David Chadwick" of "University of Salford" (in a department called "ISI") the power to let "staff" from Salford University use our printer to "print" as many "copies" of documents as they like, as we've very close working ties with them. However, the head of department is keen that no one from Salford's "Long Book Dept" be allowed to use our printer (as it would soon get too expensive). Also, he doesn't want anyone from Germany (i.e. anyone from country code "de") to use the printer, for licensing reasons. Finally, he says that the College's computing standards mean you'll have to give the policy an ID number of "1.2.3.4.5.6""

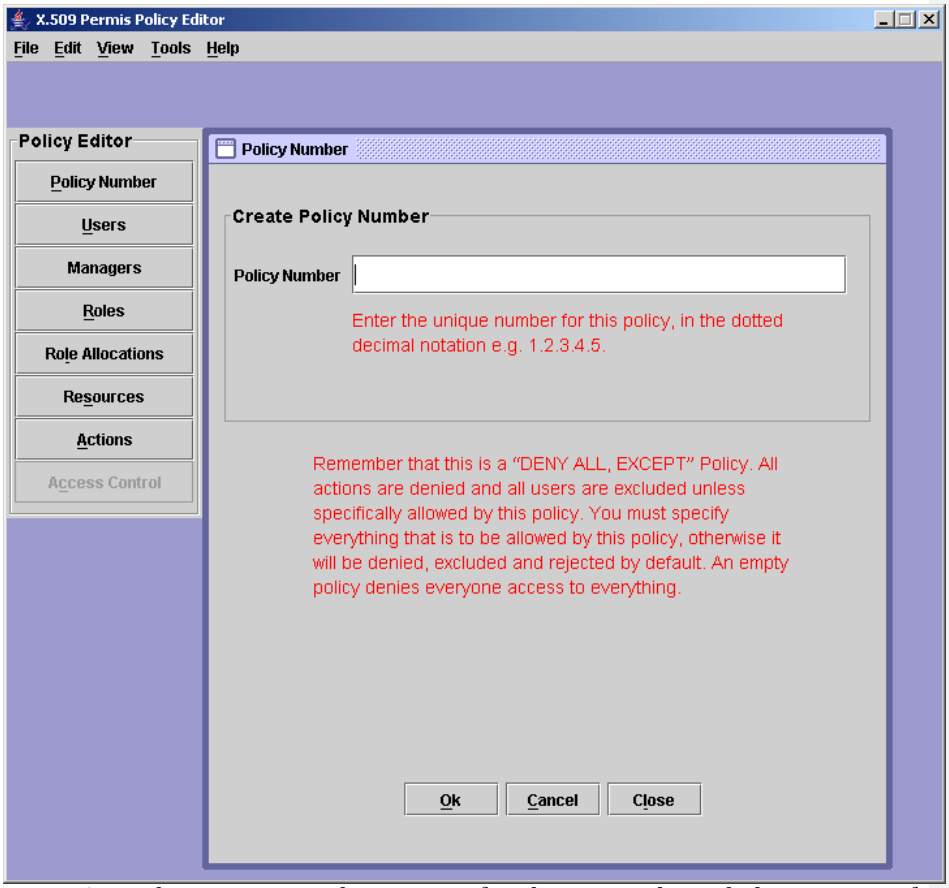

Figure 3. explanatory text in the user interface laying out the underlying nature of PERMIS policies. (Prototype 2)

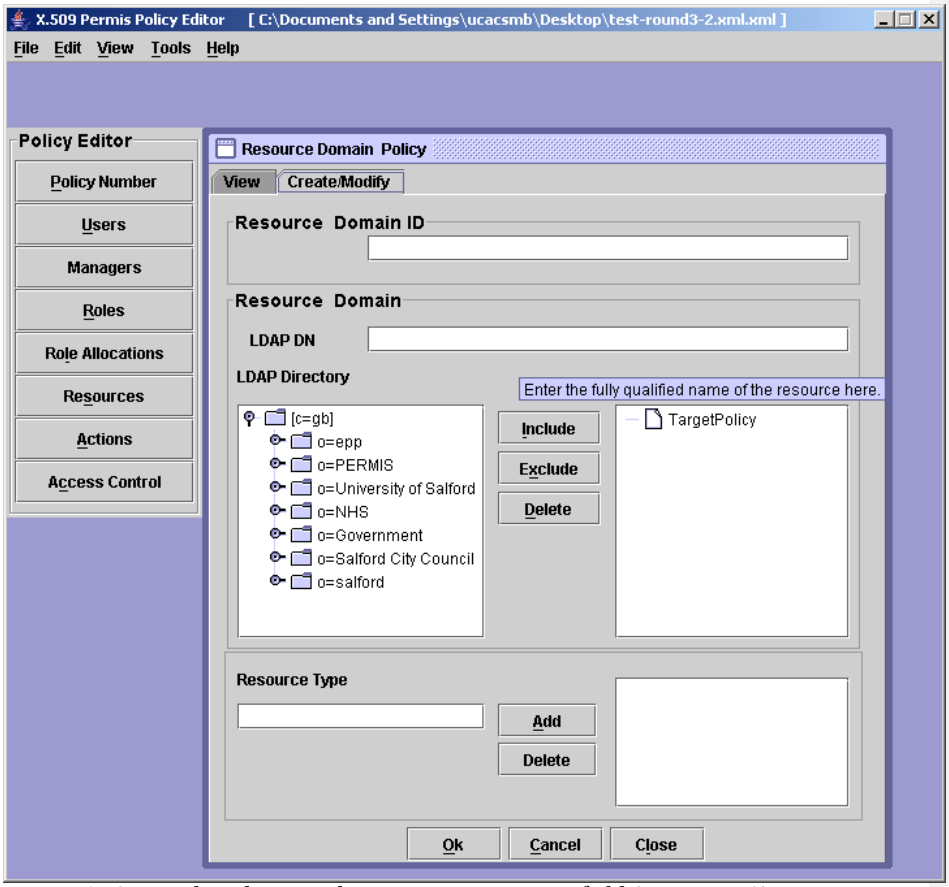

Figure 4. Screenshot showing the new Resource Type field (Prototype 2)

Table 4. Alert boxes from prototypes one and two, showing a progression to more intuitive language: a) after starting the policy editor and selecting new from the file menu, b) on clicking the okay button, c) on trying to quit without saving.

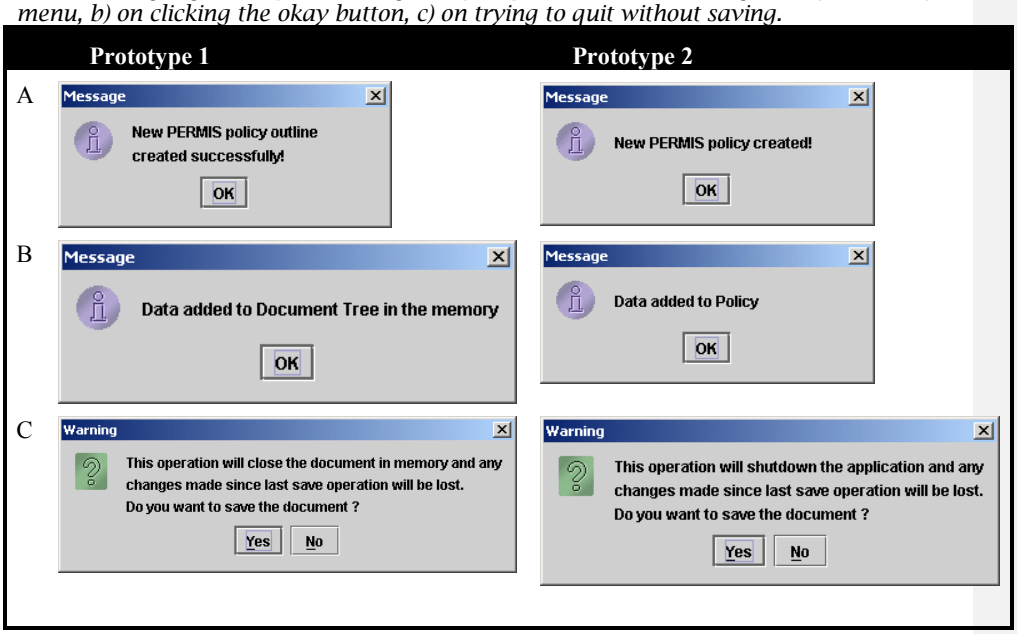

Table 5. Intuitiveness of version 2 and 3 labels. Percentage correct is the proportion of correct questionnaire responses for each label. 8 respondents answered two questions about each label.

v3 Labels	% correct	v2 Labels	% correct
User privileges	84.4%	Access control	21.9%
Account administrator privileges	87.5%	Role allocations	37.5%
Account administrators	87.5%	Managers	81.3%
User types	84.4%	Roles	56.3%
Where users are from	100.0%	Users	43.8%
My protected resources	90.6%	Resources	90.6%
Resource functionality	59.4%	Actions	37.5%
Policy object ID	100.0%	Policy number	93.8%

Table 6 – Alert box on clicking OK when there' s an empty ID field and LDAP DN field – see figure 4.

Prototype No.	Alert box
Prototype 2	
Prototype 3	

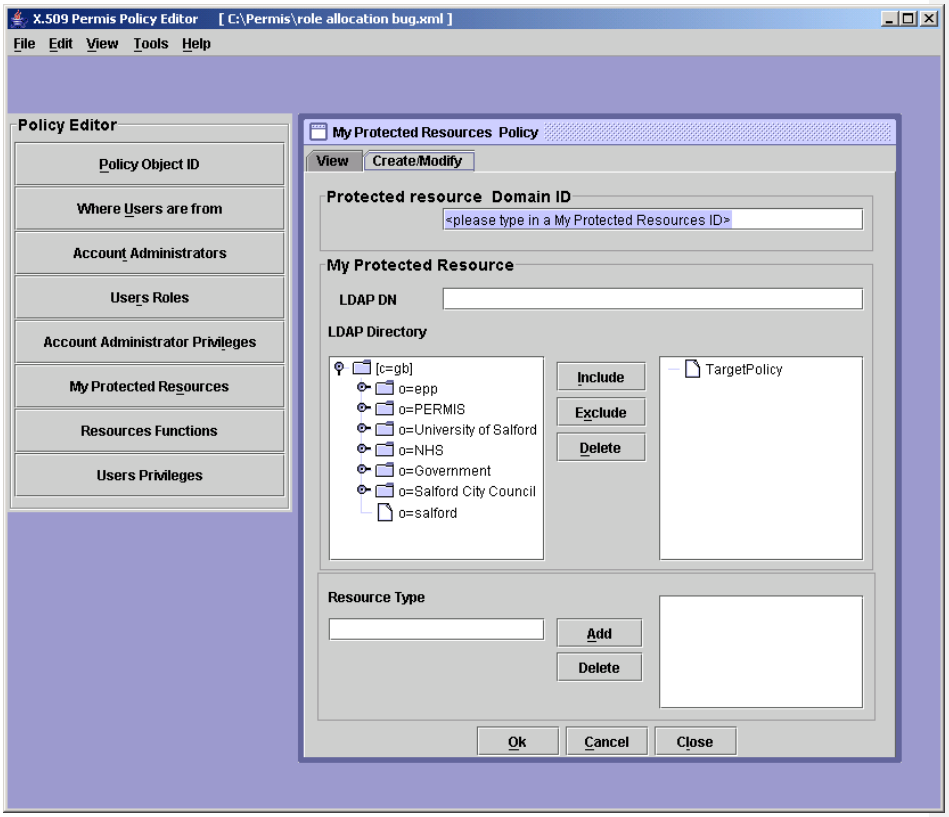

Figure 5. Screenshot from prototype 4, showing most version 4 labels and tailored ID field focus /instructions