Expressions of Expertness: The Virtuous Circle of Natural Language for Access Control Policy Specification

Philip Inglesant, M. Angela Sasse
Department of Computer Science
University College London
Gower Street, London WC1E 6BT, UK
{p.inglesant, a.sasse}@cs.ucl.ac.uk
+44 20 7679 3039

David Chadwick, Lei Lei Shi
Computing Laboratory
University of Kent
Canterbury, Kent, CT2 7NZ, UK
{d.w.chadwick@kent.ac.uk,L.L.Shi}@kent.ac.uk

ABSTRACT

The implementation of usable security is particularly challenging in the growing field of Grid computing, where control is decentralised, systems are heterogeneous, and authorization applies across administrative domains. PERMIS, based on the Role-Based Access Control (RBAC) model, provides a unified, scalable infrastructure to address these challenges. Previous research has found that resource owners generally do not understand the PERMIS RBAC model and consequently have difficulty expressing access control policies. We have addressed this issue by investigating the use of a controlled natural language parser for expressing these policies. In this paper, we describe our experiences in the design, implementation, and evaluation of this parser for the PERMIS Editor. We began by understanding the ways in which non-security specialists express their Grid access control needs, through interviews and focus groups with 45 resource owners. We found that the many areas of Grid computing use present varied security requirements; this suggests a minimal, open design. We designed and implemented a controlled natural language system to support these needs, which we evaluated with a cross-section of 17 target users. We found that the interface is highly usable for interaction: participants were not daunted by the text editor, and understood the syntax easily. However, some strict requirements of the controlled language were problematic. Using natural language helps overcome some conceptual mis-matches between PERMIS RBAC and older paradigms; however, there are still subtleties which are not always understood. In conclusion, the parser is not sufficient on its own, and should be seen in the interplay with other parts of the PERMIS Editor, so that, iteratively, users are helped to understand the underlying PERMIS model and to express their security policies more accurately and more completely.

Categories and Subject Descriptors

H5.2. Information interfaces and presentation: User Interfaces: Natural Language

General Terms

Design; Security; Qualitative Methods; Observations

Keywords

Authorization; Access Control; Grid computing; RBAC; Controlled Natural Language

Symposium On Usable Privacy and Security (SOUPS) 2008, July 23-25, 2008, Pittsburgh, PA, USA.

1. INTRODUCTION

It should be indisputable that security and usability must co-exist. As long ago as 1975, Saltzer and Schroeder [22] promoted the principle of *psychological acceptability* of security mechanisms: protection mechanisms must be easily applicable by their target users. Security which is not usable is likely to lead to errors [24] and the creation of workarounds [1], and ultimately to a reduction in security.

The arguments for usable security mechanisms are well-known even if they are not always easy to put into practice. This paper presents an effort to improve usability of a tool for a fundamental aspect of security – access control. Controlling access to resources is one of the most effective security measures, but is currently often given a low priority by resource owners because of the difficulties they find in using existing authorization methods [3]. The challenge, then, is to produce interfaces to access control tools that are accessible, and to enable resource owners to correctly set controls that reflect their security needs.

PERMIS [9] offers a basis for achieving usable access control. In essence, PERMIS is an integrated, Role-Based Access Control (RBAC) [23] infrastructure which provides all the necessary facilities for resource owners to manage authorization policies, and for these policies to be implemented in e-Science applications.

Recognising the inherent difficulties in setting access control policies, PERMIS provides a Policy Editor with several complementary interfaces. The earliest interface was a Graphical User Interface (GUI), with tabs and drop-down menus. Later, a wizard for creating new policies and a policy tester were added.

These interfaces successfully reduce the burden of maintenance of large and complex policies, but a vital aspect of policy specification is to ensure that the resource owner avoids mistakes arising from basic misconceptions [8]. To some extent, this need can be met by matching the language of the Editor to that of the target users [18]; earlier work using Conceptual Design enhanced its usability of the GUI [5]. However, we also realised that although careful design of an interface can help resource owners to understand *what* has to be done to write access control policies, they still have to work out *how* to state those policies correctly.

The new PERMIS user interface presented in this paper takes a different approach: it uses *controlled natural language* to reduce the "*distance*" [19] between resource owners' familiar, real-world access control needs and their expression in computer terms. This is not a replacement for the older interfaces, but is complementary to them. It aims to "*match the users' world*" [18], not by incorporating their language into an interface which still reflects the underlying computer logic, but more fundamentally, by enabling resource owners to express policies in their own natural ways of thinking.

The remainder of this paper is organised as follows. Section 2 reviews previous research in usable security policy specification and identifies key issues in Grid and RBAC authorization. Section 3 describes the first phase of our work to address these issues, in which we gained an understanding of the ways in which resource owners express their access control needs, and used these findings in the design of a controlled natural language interface. We evaluated the usability of our interface in scenario-based observations as detailed in section 4. In section 5, we relate our results from this evaluation to the key issues identified in section 2. We conclude by considering ways in which usability of our interface might be improved and give brief pointers for future work.

2. BACKGROUND: FROM USER VALUES TO ACCESS CONTROL POLICIES

The overall problem which this paper addresses is that it is often difficult for resource owners to bridge the gap between their security needs, which might be understood in quite general terms, and the expression of those needs in concrete, computer terms [14]. This problem has been addressed by the usable security research community over the past 10 years; in this section we review the key previous work.

The application domain and target user community for our work was Grid computing; we address problems which have been found in the specification of Grid authorization policies [5]. We consider ways in which resource owners' natural expertise can be engaged, and we show that controlled natural language has been used in similar areas and is a good candidate to enable the expression of access control policies in an intuitive way.

We conclude our review of the background to our research by identifying the usability challenges of policy specification in our underlying PERMIS RBAC authorization model.

2.1 What the Resource Owner Intends

It is important to clarify the direct, but often obscure, path from the intentions of resource owners through to low-level actions by IT systems.

Resource owners (including developers, system administrators, end users, and others) generally have good knowledge of the assets under their control and of who should be allowed to do what [12]. This is at the level where, if asked whether person A should be allowed to use a resource X, most can answer "yes" or "no", based on their knowledge about the person and rules about how the resources should be used. The problem is not, then, that resource owners lack knowledge of their access control needs, but that they may have difficulty in "programming" them correctly in an authorization system [14].

Moreover, emerging security needs must work in a context very different from that for which security paradigms were designed. In contrast to a conventional mainframe system, where security was essentially under the control of a single system administrator, today it is often required to secure resources on a decentralised network with no single point of control. Security policies may need to account for new parameters, such as the location of a requestor [15]. Whilst the concept is easy to understand, the parameters are often difficult for resource owners to express in the computer security policy language, particularly if, as is most often the case, they do not have security expertise [14].

2.2 Authorization Reaches Out

In practical implementation, these emerging paradigms can only be made tractable with a clear understanding of the differences and interplay between *authentication* and *authorization*. Authentication is the process of determining and verifying the identity of the user (or other actor) making a request, whilst authorization is determining whether to grant a user (or other actor) a particular form of access to a resource [22].

In practice, authorization is far more important than authentication, but, perhaps paradoxically, authentication has, to date, been studied in more depth. This could be because usable and fully-verifiable authentication systems are a prerequisite for authorization, which depends upon them in order to function correctly. Authorization and authentication are inter-dependent; privacy invasions, for example, can result from designers' (and implementers') inability to foresee how, or by whom, data might be used [2]. But authorization presents its own usability issues. In this paper, we focus on the usability of the interface for setting the authorization or access control policy.

Traditionally, access control – whether a policy-based model or lists on each resource - has not been controlled directly by "end users" of the system, but rather by system administrators. Increasingly, for example in WebDAV or NTFS security [6], end users, as resource owners, are indeed responsible for setting the access controls. There is, then, not always such a great distance between the people who set access control policies and those who are subject to them. This reinforces the observation of Yee [25] and others that responsibility for setting and maintaining security controls often falls to non-specialists, who are primarily concerned with other more immediately pressing tasks. A usable way to express access control is essential if it is to be followed reliably, but there is the additional danger that users may not fully understand the implications of their security actions.

2.3 Authorization in Grid computing

Setting access controls in ways which are comprehensible and clear for non-specialists is all the more important in the growing area of Grid computing. Here, the systems being protected, and the applications running on them, are heterogeneous, and include very expensive or highly confidential resources. Grids may expand to very large computer systems, potentially accessed by many users or by other computers. This large, complex network of actors, resources, actions, permissions, and constraints leads to a correspondingly dynamic and complex security configuration.

Moreover, because the computers in a Grid are spread across administrative domains, a resource owner will usually not grant access directly to an individual known to him or her. Conversely, with the use of schedulers, a person requesting use of Grid resources might not know in advance the particular set of hosts on which his or her request will be actioned [16].

Applying these kinds of complex configurations resembles end-user programming rather more than it resembles interactions which are commonly performed using a GUI. To the extent that this is a form of programming, it is a process of transforming a conceptual plan "in the head" of the user/programmer in familiar, informal terms, into a form which is compatible with a computer. There is long-standing empirical research into how non-programmers "*naturally*" think about programming [19]. More recently, Rode, Rosson, and Quiñones [21] have made a study of how non-programmer webmasters think about some common processing needs in web applications. They discovered many *mis*-conceptions and unconsidered assumptions; of particular relevance to this present research, they found that non-programmer webmasters can usually devise a simple authorization scheme, but it is almost always incomplete.

2.4 Language and Human Intentions

The same authors who have pointed out the distance between users' mental plans and the expression of those plans in terms which are compatible with a computer have also suggested natural-*like* language as a way to decrease this "distance" and increase the "naturalness" [19,21]. Pulman [20] suggested that controlled natural language might be a way to enable experts in some domain to express their expertise in a way which could translated into a portable, computer-readable form. The problem of expressing access control policies is similar, but is more immediate since information systems are growing rapidly in complexity, with consequent access control challenges.

Natural-*ness* is not necessarily best achieved by a full natural language [19]; *controlled* natural language is not in any way a compromise. Controlled language can be tailored for specific uses, such as web service protocol descriptions or the construction of ontologies, as has been done by the General Architecture for Text Engineering (GATE) team in the Semantic Knowledge Technology (SEKT) project[1]. Significantly, these have compared favourably with a GUI-style ontology editor [13].

The use of controlled languages has been found elsewhere to be highly usable for end-user specification of security and/or privacy policies. SPARCLE is designed for natural expression of privacy policies [4]. Adage includes a formal logical language alongside a GUI for the expression of RBAC policies [26]. However, the motivation and approach behind PERMIS is quite specific. We have already seen (section 2.3) that Grid computing presents particular authorization challenges. PERMIS is designed to address these challenges on the basis of RBAC. As the following subsection shows, RBAC provides a means to make Grid security manageable, but also presents new conceptual difficulties for non-specialist users.

2.5 Challenges in RBAC Policy Specification

Access control in PERMIS is based on a well understood, unified access control model, a variant of RBAC. One of the advantages of such an authorization policy is that, unlike access control applied at the level of each subject or each target resource (such as access control lists or Unix-style read-write-execute), a unified policy is therefore more maintainable and more scalable [7].

As well as providing a solid foundation for security implementations, RBAC is applicable to completely general situations, rather than being drawn from the privacy or security needs of particular application domains. This is one of its strong points, but without careful interface design this could place a correspondingly greater onus on the resource owner.

Unfortunately, it has been found that resource owners generally are not familiar with RBAC, and consequently have a partial understanding of "*what-needs-to-be-done*" [5] to implement access control; this would lead to policies which are incomplete or which contain unnecessary elements. In particular, Brostoff, Sasse, Chadwick, Cunningham, Mbanaso & Otenko [5] identified two classes of problems which they labelled the "*policy components*" and "*policy paradigms*" problems.By "*policy components problem*", they mean that resource owners do not understand some of the basic structure of the PERMIS RBAC policy space, such as Subject Domains (the domains from which users can be allowed to access resources) and role assignments around Source of Authority (SOA) or domain administrators.

By "*policy paradigm problem*", they mean that resource owners are unsure which objects should be included in a policy, and which left out, if they follow the mental model of traditional access control such as "*explicitly grant and explicitly deny access*". Because PERMIS RBAC policies exclude by default all permissions which are not explicitly granted (the *deny all access except* model), policies are likely to be inefficient rather than non-functioning or insecure, or might unnecessarily deny access to groups of users who should have access but who were not mentioned in the policy. Nevertheless, there is a risk of unintended outcomes whenever a resource owner does not understand a key aspect of policy specification.

There are, then, two sources of risk arising from mistakes in setting access control policies. Mistakes can be due to the complexity of the policy, with a consequent likelihood of omissions, ambiguities or inconsistencies [8]. There can also be mistakes which follow from a basic mis-understanding of the underlying security model, as Brostoff et al. identified [5]. These new classes of "programming" error do not, of course, diminish the possibility of simple "slips", lapses or spelling errors, which also have to be eliminated if the policy is to function as the user intends.

For the reasons we discussed above, we believe that constrained natural language can overcome problems for users of knowing "*what-needs-to-be-done*" and can enable "slips" to be easily detected. At the same time, once conceptual shortcomings have been addressed, users still need to be supported to know *how* to use the interface to express their policy. The challenge for us, then, was to allow resource owners to express policies without requiring them to have any specialist knowledge of RBAC or access control models, and to design an interface which is usable in this more conventional HCI sense.

3. EASY EXPRESSION OF AUTHORISATION POLICIES

This was the point of departure for the Easy Expression of Authorisation Policies (EEAP) project. As part of the PERMIS infrastructure, EEAP is particularly concerned with security issues in e-Science, Grid computing, and web services generally.

[1] http://gate.ac.uk/; http://www.sekt-project.com/

3.1 The Virtuous Circle of Authorization Policy Specification

The fundamental idea underlying EEAP is the *virtuous circle* of expressing authorization policies, a concept developed particularly to support resource owners in Grid computing throughout the entire process of policy specification [8]. The *virtuous circle* is based on the realisation that language stands in a special relationship to human understanding. GUI visualisations, from this viewpoint, are complementary to natural language, rather than being the only means of reporting the meaning of access control policies. The user can choose either, or can switch between the two, so that checking is completely available for both visual and linguistic cognition.

We started the project with the natural language output for the *virtuous circle* already in place, as part of the PERMIS Editor GUI. Policies expressed using the GUI or a Wizard, are transformed into machine-processable form in XML according to the PERMIS DTD [7]. The XML is then transformed (using an XSL stylesheet) *back* to the user as natural language. The final policy is therefore available to the user in three forms: raw XML, the familiar GUI screens, and the natural language output. Crucially, the natural language display also shows diagnostic error and warning messages, a point to which we shall return later. Because the output, in whichever form they prefer, is generated directly from the computer-readable form of the policy, the user can be confident that it reflects the authorization that will actually be enforced by the system.

In a paper published at the beginning of the PERMIS Natural Language development, Chadwick & Sasse [8] assumed that completing the virtuous circle by enabling the use of controlled natural language *input* of security policies as well as for their display would greatly reduce the scope for misconceptions of the sort discussed in the previous section, and would enable "slips" to be more easily detected. However, at that early stage, this remained to be investigated empirically. In this paper, we revisit these assertions, in the light of our experiences with applying these ideas in practice.

3.2 Grid Security in the Wild

The process of developing a natural language input for the PERMIS Editor began by interviewing 45 e-science practitioners across the range of e-science application areas: the hard sciences, medical research and bioinformatics, earth sciences, and arts and humanities. Interviews were semi-structured, with an average length of about 45 minutes. They were voice-recorded for transcription using Grounded Theory [10]. 18 participants were interviewed individually and the others in small focus groups (2-4 participants).

This first phase of the research had three main purposes:

1. To understand the major requirements in Grid security, and how they are *expressed* by Grid resource owners;
2. To inform the design of the ontology which underlies PERMIS access control policies and is the first stage of natural language processing; and
3. To inform suitable scenarios for the later evaluation of the natural language interface.

3.2.1 Grid Security: Varied Uses, Complex Needs

From the interviews, it was evident that Grid security policies are hard to specify, not only because access control is not well understood by resource owners, but because real-world situations are complex and changeable.

Grid computing has varied and sometimes incompatible requirements: access to large volumes of data, fine-grained access control, making specialised data or software widely available to the research community, providing very high-powered computer processing, and maintaining the confidentiality of data. Data integrity is always important, but especially where data volumes are very large. In some areas, for example some kinds of humanities data, there are commercial considerations; data may be restricted because it has gained a high commercial value in electronic form, even if it is public data. Conversely, the availability of electronic images of artefacts such as rare documents may remove restrictions imposed by the physical vulnerability of the originals.

3.2.2 R-what? Implications for Ontology design

We already knew that knowledge of RBAC is not widespread among Grid users, apart from security specialists. The findings from the interviews re-enforce this, but, in the absence of easily specified security policies which fit their needs, resource owners are adopting simpler, all-or-nothing policies.

The means for expressing and maintaining access control policies must be flexible enough to handle very different needs in different applications, while remaining comprehensible by the intended users. Our original intention was to extract security terms (words and phrases), synonyms, and antonyms, and relate them to the model formalized in the ontology. However, our findings from this first phase suggested the need to keep the ontology as general as possible by defining only the basic classes and sub-classes, avoiding application-specific instances.

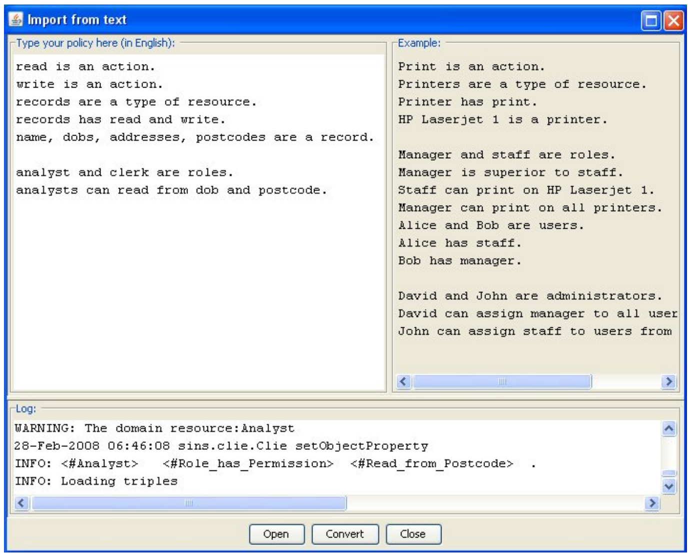
3.3 Putting the Virtuous Circle into Practice

Underpinned by our ontology, the last link in the chain of a *virtuous circle* of authorization policies has been now put in place: the Policy Editor supports controlled natural language [20] input of the most essential features of the RBAC model. It is now possible to express an access control policy in controlled natural language, to have this transformed in to XML, and for this to be re-presented back to the user as a diagnostic display, in natural language or another form of his or her choice. However, the parser does not yet include the full functionality of the PERMIS RBAC specification.

The constrained natural language interface provides a simple layout. On the left-hand side, the user types sentences, each of which represents a rule in the natural language form of the policy. These sentences do not have to correspond to sections of the final computer-readable policy, but are in any order which makes sense to the user. Indeed, rules can be combined using comma-separated lists:

> Manager, owner, and clerk are roles.
>
> Managers and owners can print on LPT1 and HP Laserjet.

The space for entry of the natural language text is a simple editor, with functions such as cut/copy and paste and insert or over-write, and shortcuts Ctrl+X, Ctrl+V, Ctrl+C (Figure 1). The right-hand side of the same window shows an example of a policy in constrained natural language. This is a key part of the interface; resource owners should be able to express security policies guided by a few example rules and only minimal other guidance.



```
read is an action.
write is an action.
records are a type of resource.
records has read and write.
name, dobs, addresses, postcodes are a record.

analyst and clerk are roles.
analysts can read from dob and postcode.
```

```
Print is an action.
Printers are a type of resource.
Printer has print.
HP Laserjet 1 is a printer.

Manager and staff are roles.
Manager is superior to staff.
Staff can print on HP Laserjet 1.
Manager can print on all printers.
Alice and Bob are users.
Alice has staff.
Bob has manager.

David and John are administrators.
David can assign manager to all user
John can assign staff to users from
```

```
WARNING: The domain resource:Analyst
28-Feb-2008 06:46:08 sins.clie.Clie setObjectProperty
INFO: <#Analyst>   <#Role_has_Permission>  <#Read_from_Postcode>  .
INFO: Loading triples
```

Figure 1: The controlled natural language interface

3.4 PERMIS Controlled Natural Language

It is important to understand that this is *controlled* natural language processing. We have already shown that it is natural-*ness*, not natural language in itself, which is of interest as a means of reducing the distance between users' intentions and their formal expression. From the implementation point of view, natural language is ambiguous and complex, and consequently very hard to process and translate by machines, and such tools that do exist are usually not freely available. Controlled natural language, in contrast, provides a strictly limited vocabulary and/or grammar. This makes machine processing much easier [8], while still being tailored to the specific requirements of the interface.

Our natural language processor has made use of a GATE implementation, Controlled Language for Ontology Editing (CLOnE), itself built on earlier work of the GATE team, Controlled Language for Information Extractions (CLIE). Details of the ten syntactic rules of CLOnE are given in [13].

In effect, specifying an authorization policy is very similar to defining an ontology. The critical point, though, is that this is transparent to the user.

3.4.1 The Controlled Natural Language Interface

Some powerful usability features of CLOnE have been carried over into our natural language interface. For example, the parser can identify matching nouns differing in singularity or plurality, and can handle irregular forms or non-English loan words ("*There are Children. Xavier is a child*"). This feature is further enhanced in our implementation, which is more lax than natural English in terms of grammatical agreement of singular or plural of subject and verb ("*supervisors and office staff are an employee*" is acceptable even though it is incorrect English).

In our constrained natural language, we have extended CLOnE in four respects which pertain specifically to authorization policies:

1. A simple way, using triples, to allocate permissions to roles: <Role> can <Action> on <Target>; for example, "*Staff can print on HP Laserjet 1.*";
2. Linking the special "can assign" permissions to role/attribute administrators: <Admin> can assign the <Role> to <Subject>; for example, "*David can assign the manager role to Alice.*", or "*John can assign the clerk role to users from department A.*";
3. Using "trust" as a variation of 2) in "I trust <Administrator> to say who <Role> are; for example, "*I trust David to say who managers are.*";
4. Conditions to constrain a permission: <Subject> can <Action> on <Target> if <Condition>; for example, "*Staff can print on HP Laserjet 1 if copies is less than 10.*"

New rule 4) is only partially implemented in the current early version of the language parser and so it did not feature in the evaluations.

3.4.2 Classes and Instances: New Entities from Old

Grouping of types within categories is a suggestion of Karat et al. [17] as a means to overcome scaling issues. Our natural language parser provides two useful grouping features, which allow users to refer concisely to properties which apply to the whole group.

When an entity class is created, a special pseudo-instance is automatically created at the first time, called "*all_<class>*" (eg. "all_Printer"). This can be used later to define properties of every object of that type (every instance of the class). For example,

> There are printers.
>
> Managers can print on [all] printers.

A related feature, native to CLOnE, is that entities can be created as a "type of" some already existing entity (as a sub-class of a class). The sub-class inherits the properties of the super-class; for example, a type of *resource* inherits the *actions* property.

3.4.3 Language is Parsed in Context

This ability to create new types of entity from existing ones is used in PERMIS so that the process of specifying a policy does not start from an empty ontology, but builds on a small set of hidden and pre-defined classes and relationships. The user is, in effect, creating instances of classes and defining new classes from existing ones, but is unaware of the inbuilt definitions. This means that the sentences written by users are parsed in the context of an access control policy for which the outline is already pre-loaded.

This context is also in the form of natural language with exactly the same syntax:

> There are users, roles, resources, actions, parameters and permissions.
>
> Resources are also called targets.
>
> Users have roles.
>
> Roles have permissions.

Permissions have resources and actions.

Resources have actions.

…

These rules, which describe the underlying RBAC model, are loaded and parsed before any user input, to build an ontology model with pre-defined classes and relationships from the authors' background knowledge of RBAC and PERMIS.

This background context removes from the user the burden of defining from scratch the ontology of the RBAC model. But a more important purpose of the context is to align the security model in user's mind with the RBAC model used in the computer system. There might be a different model in their mind; by providing a pre-defined RBAC model of Roles, Permissions, and other elements, the user is enabled to specify a policy in these terms and with their own variations, which can then be exported to the final computer-readable format.

It is important to emphasise that users are not expected to know, or to need to know, anything about the underlying ontology, RBAC model, or rulesets; we discuss these here only to clarify the connection between natural language and the final policy expression. Classes, properties, pre-defined elements, and the relationships between them, and from them to the final policy, are transparent to the users; they need only to learn a few simple rules and follow the example text.

4. EVALUATION

In the first phase of the research we interviewed a wide selection of e-science users and administrators, used this as the basis for requirements and for enhancements to the ontology design, and implemented a controlled natural language interface to reflect the requirements and the ontology. We now turn to the evaluation of our interface.

From the review of previous research and our beliefs outlined above, we derived four research questions:

1. Overall usability: can target users understand the syntax of the controlled natural language, using the example?
2. Can target users understand the "building blocks" of a PERMIS policy (resources, actions, roles, and administrator and role assignments)?
3. Can target users avoid misconceptions in the RBAC/PERMIS model when using the PERMIS natural language editor?

And, finally, the overall question:

4. Using controlled natural language, with the simple examples provided, are target users able to specify policies accurately, reflecting their real-world intentions?

This is a quite specific understanding of usability, tailored to the needs of access control specification. At this stage, we did not attempt to measure other aspects of usability, such as subjective satisfaction or efficiency. We chose a scenario-based approach, recorded and observed in a controlled environment.

The first scenario (Figure 2) was designed to reflect common real-world access control needs without making reference to any particular field of application. Where time allowed this was followed by a more complex scenario; for participants with prior e-science experience, this second scenario was drawn from their field of work, based on the interviews conducted in phase one; for others, the second scenario was a variant of the first.

These scenarios were quite specific in terms of access control, but in a form which could not be simply entered verbatim into the natural language processor. In taking this approach, we assume that real-world users know what they want to control; our interest is in their ability to express their intentions. This requires a careful methodological balance between the need to be clear about what the policy should say, and the risk of simply giving users a set of words they can copy.

4.1.1 Participants

Seventeen participants were recruited in three complementary ways: using internal email lists; a request to IT-related staff working internally in the college library; and from a database of e-scientists built up during earlier phases of the research.

Target users are e-scientists (researchers with knowledge of their research domain and some understanding of Grid computing), senior research management (Principal Investigators) and administrators (possibly departmental administrators or information systems staff). Although they have good computer skills, they are not computer security specialists. All of the participants were fluent in written and spoken English, although not all had English as a first language.

All 17 participants were from our target group of users; all were highly computer-literate and working in a variety of computer-related areas. They are target users, even though not all of the participants had specific e-Science experience. This reflects likely usage in the real world, where security policy authoring is often handled by team members who are not e-science or security specialists.

Participants included 7 e-science researchers in Earth Sciences, Medical, Crystallography/Chemistry, Physics, and Arts & Humanities; and 10 participants without specific e-Science experience, of whom 4 were computer science researchers and 6 library computer professionals (web and database administrators, project managers).

4.1.2 Conducting the observations

We already know that resource owners, as non-security specialists, usually do not have any formal understanding of access control models, particularly RBAC [5]. However, we believe it is realistic to expect that resource owners would have informal knowledge of basic access control concepts, perhaps from the PERMIS Editor tutorial which new users are encouraged to follow. To ensure that these basic ideas were understood, rather than asking participants to complete the tutorial we prepared a short (1 page) description of the basic RBAC concepts.

This was read to them by the experimenter, rather than asking participants to read it for themselves; this was to overcome different abilities in grasping written information and to allow the experimenter to check understanding at key points.

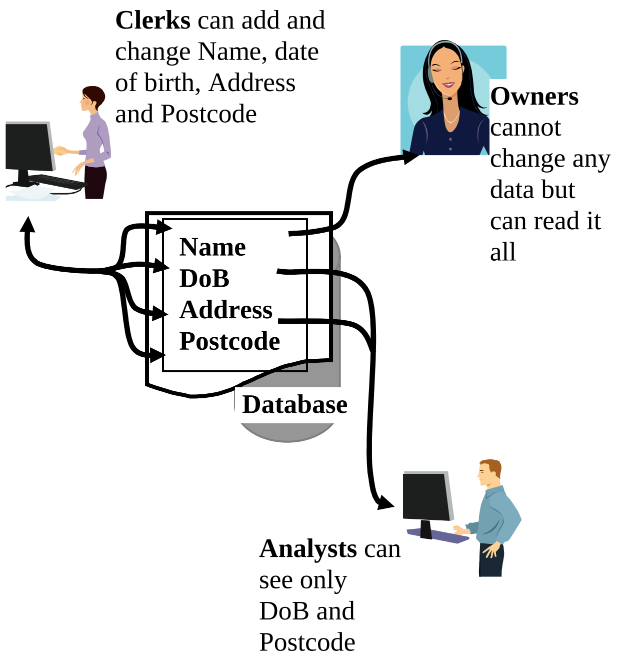

Clerks can add and change Name, date of birth, Address and Postcode
Owners cannot change any data but can read it all
Name
DoB
Address
Postcode
Database
Analysts can see only DoB and Postcode

Figure 2: General scenario 1 (diagrammatic form)

Each participant was then given, in printed form, the first scenario presented in two formats, as a simple written list of requirements and in diagrammatic form (Figure 2).

To reflect what we believe to be the common point from which policy authoring starts, we presented participants with scenarios as both words and a diagram. We hoped the participants would work mainly from the diagram. However, we found that in practice, they mostly ignored the diagram and worked from the verbal description; in future, we would use diagram-only for similar scenarios.

Participants were told that they could take as much time, and as many attempts as they needed to complete a scenario and to produce a working access control policy. (In practice, with two exceptions, the sessions were limited to one hour.) Two general scenarios were prepared without any reference to a specific application area. We aimed to avoid any real-world references, to avoid participants making assumptions about levels of seniority or other aspects.

The first, simpler, scenario contained three roles, three resources with three possible actions on them, and one administrator. The second scenario was only a little more complex, adding the concept of users' domains. The scenarios were phrased to include concepts which are not normally expressed directly in RBAC: access denials; access to "all" instances of a resource type; and "groupings" – different elements which are specified as being of a type, as well as "background" elements such as a database containing the resources.

Interactions, every action on the screen, keyboard and mouse, as well as voice, were recorded using Camtasia Studio[2]. We did not use a formal think-aloud protocol because this can be distracting, but we did encourage participants to make comments, and occasionally the experimenter would ask a participant to explain an action. These comments and questions were noted during analysis and form a valuable input to the results.

[2] http://www.techsmith.com/camtasia.asp

4.1.3 Analysing the Observation Data

The analysis proceeded as follows. Each of the recordings was replayed as many times as necessary, with the analyst noting in a spreadsheet the times at which key events occurred, and each time the participant clicked "Convert"; this is considered to be a "try".

Measured times include the time taken for the participant to read the scenario, but not the time taken for the observer to read out the background description of basic RBAC concepts. We call this the elapsed time since "handover", the point at which the observer finished reading the introduction and RBAC overview and explicitly made clear to the participant that the observation was now under way.

We expected the participants to continue until a workable policy was produced, within the time constraint of one hour overall. Therefore, rather than a metric for scoring rules, our measure of the accuracy of policy specification is the number of "tries" made by each participant. This needs to be considered in conjunction with the overall time, since some participants chose to correct errors themselves, before clicking "Convert".

At the same time as recording the timings and number of "tries", the analyst noted significant questions and comments by the participants, used in the qualitative analysis which follows.

5. RESULTS

5.1 Overall results

Overall the results are encouraging: all 17 participants grasped the basic concept of expressing policies in controlled language without difficulty. The time taken and number of attempts to produce a complete working policy in the first scenario was higher than we would like in real use (average 35:45 minutes and 6.6 tries), but we expect that this will fall as users learn the simple grammar of the constrained language, and as they re-use and amend existing "scripts".

We now address in more detail the questions raised at the beginning of section 4.

5.2 Usability of Controlled Natural Language

The constrained vocabulary and the names of objects in the predefined ontology (resources, actions, roles, permissions) are well understood. Participants did not need to understand the relation between verbs such as "can" or "assign" and the creation of entities in the ontology in order to specify workable policies. Some participants considered the language almost as a "script", using that term in feedback to the observer.

5.2.1 Usability of the Editing Space

Our first concern in overall usability was that presenting participants with an almost empty space on which to type, with minimal editing controls and only an example text as a guide, might be daunting.

However, this does not appear to have been a problem for our participants. Measured as the time taken between the "handover" of the session and the participant starting to type on the text editing space, this was average of 4:20 minutes. We believe that this is a sufficiently short time, including the time to read the instructions and scenario, to indicate that participants were not daunted by the emptiness of the screen.

The majority (15 of 17) of the participants were able to specify an accurate, workable access control policy for at least the simple scenario within 45 minutes and 10 tries. Excluding the two outliers, mean times for completion of the first scenario fall to 26:45 minutes in 5.125 tries.

We hope that the overall times will fall as target users learn the requirements of our constrained natural language. There is some evidence to support this. Of the 7 non-specialist participants who proceeded to the second, slightly more advanced, scenario, the mean time was 12:36 minutes in 2.5 tries, significantly better than the first. However, note that this second scenario was conducted immediately after the first, and was very similar, adding only two additional user administrative domains.

5.2.2 Usability in Specifying Policy Elements

It is clear from the detailed timings that some task elements are more readily understood than others. Adding the three roles, Clerk, Owner, and Analyst seems to have caused little problem. Similarly, almost all participants managed to say "*John is an administrator.*" on the first attempt.

There is a specific issue which caused some problems; this relates to accuracy, and is also a part of the general usability of the interface. This follows a design feature of the controlled language: it is strict with regard to the pre-definition of entities. References to entities do not cause that entity to be created; if it has not been defined earlier in the policy, then this is an error.

This is by design; it applies to all entities – resources, resource *types*, roles, actions, users, and permissions; the aim is to prevent mistakes introduced by typing errors. For example

> Clerk can read *databsae*"

will be reported as an error, rather than creating an incorrect resource instance "*databsae*". However, this does, naturally, add to users' workload by requiring that each instance must be explicitly defined before it can be referenced.

5.2.3 A Parsing Problem: Prepositions

The quantitative data does not show *why* some of the rules proved difficult to specify. But analysis of the qualitative data shows one of the most common problems: forgetting to add prepositions between verbs and the corresponding object:

> Owners can read Name.

instead of

> Owners can read from Name.

It might be expected that this would be more of a problem where the verb usually does not require a preposition in natural English than with other verbs, but our observations do not support this.

Part of the scenario required a combination of write/add/change – the scenario said:

> Clerks can add and change Name, date of birth, Address and Postcode

- given like this, without prepositions. *Change* and *add* do not normally have prepositions, so the parser requires some slightly "un-natural" English such as "Clerks can change on Name ...".

The need for prepositions is a feature of the parser which would require a deep re-design to change; the appropriate design response is to guide users to follow it correctly. This is not a fundamental issue in the gap between users' intentions and their expressions of them in language. The problems found here are, however, suggestive of the ways in which participants make use of the example text. As we describe in the following subsection, a more critical incident led us to change the text slightly, to remove a more fundamental problem.

5.2.4 The Importance of the Example

An early version of the example text showed a sentence specifying a parameter for an action:

> Print has Pagenum

That is, the *print* action can take a *pagenum* parameter.

One participant attempted to specify a policy in which the resources to be acted upon were given as parameters to the action, for example:

> Write with Address.

where Address is a parameter of the Write action. Superficially, this seems reasonable, since actions can have parameters; however, in our ontology it is not possible to restrict access according to the parameters to an action – properties apply to classes/subclasses and their instances, that is, to objects, not parameters.

This line in the example text was removed from the example text for later trials, and, not surprisingly, no further participants attempted to express the policy in this way. Our point here is not that using parameters is or should be incorporated into the ontology as a way to control access; the point is that this incident throws light on the use made by participants of the example text. However, in a later subsection we give a contrasting example.

5.3 Specifying Policies in Terms of the Ontology Elements

We found that participants had little difficulty in understanding the basic elements, the pre-defined entities which are the "building blocks" of an RBAC policy: roles, actions, and resources. This in itself is a positive result, since RBAC revolves around these concepts, which are unfamiliar, as access control elements, to most of our participants.

We did, however, identify two common classes of problem in the use of these pre-defined building blocks. These both concern users' conception of elements of the underlying ontology, and so present a design question about how best to guide users, without explicitly exposing the design of the ontology.

5.3.1 Understanding the policy "building blocks"

Participants should not have to know about the pre-defined entities (ontology classes – see section 3.4.3), which are the "building blocks" of an access control policy. However, they do need to understand that, although they are free to define the names of new entities (classes and instances), these new entities must be defined in terms of the existing entities.

For example, in order to say "*dirac is a computer*", it is necessary for "computer" to be previously defined as a class, or, more workably, as a sub-class of *resource*:

> Computers are a type of resource.
>
> Dirac is a computer.

Some participants attempted to use this approach to define records or fields in the database as instances, but defined them instead as a *type* of resource, that is, as a subclass in the ontology, rather than as *instances*:

> Postcode is a type of resource.

instead of

> Postcode is a resource.

This could point to a more fundamental problem, which is that users do not appreciate the difference between classes and instances, which can be and often are used interchangeably. For example, in a different access control policy, Postcode might be a class; *AA1 1BQ* could be an instance of this class. This point is further illustrated in the next subsection.

5.3.2 Aggregating Elements at too High a Level

The "grouping" feature of our natural language, described in section 3.4.2, is a powerful tool for users but also presents users with the possibility of unproductive choices. In the scenario, fields which are the subject of access control are said to be within a database. This is best considered as "background", not specified in the policy; it is the object to which the policy as a whole *applies*.

Several participants attempted to define the database as the basic resource. This approach does not lead to a workable access control policy. Although grouping is suitable for entities which share common properties, the level of grouping needs to be sufficiently fine-grained that some useful statements can be made about the commonalities. If the boundaries are too broad, then users may find that they give away more authority than they intend.

Conceptually, we believe that both of these cases suggest that users are confused by the subtle distinction between objects which are naturally understood as one object (one instance), and objects which actually represent many objects (which could be classes or instances, depending on the circumstances). Example of the former: one named computer (eg. "Dirac"), a database, a piece of hardware. Example of the latter: a set of records, fields in a database. Each of the latter has many (unquantified) instances, but in access control terms, permissions are (implicitly) to the *set* of objects. Yet in the controlled language, this set of objects must be specified as an *instance*, since permissions are on instances and not on types (classes), which are regarded as being empty in the ontology. To handle this, we introduced *all_instance*s (section 3.4.2): actions can be given to all instances of a resource class.

The question of how to bridge the subtle distinction between classes and instances for non-specialists raises difficult usability issues and requires further study.

5.4 Overcoming Misconceptions

The third question concerns the usability of controlled natural language in overcoming users' misconceptions about the access control model

Notably, the participants showed no confusions around the *target domain/subject domain* distinction which was a source of misconception for Brostoff et al [5] (the first aspect of their *policy components problem*); it becomes intuitively obvious that DepartmentA is the domain from which requests originate, the *subject* domain, in statements such as:

> John can assign Analyst to users from DepartmentA.

This is despite some problems in practical application; the syntax at this point is strict, and spaces in particular seem to cause problems.

However, concerning another point of misconception identified by Brostoff et al. [5], the function of domain administrators and the separation of roles from end-users, the evidence is less clear.

5.4.1 Understanding Roles and Assignments

This second aspect of policy components concerns the special *Role Assignment Permission*, and the associated action "<administrator> can assign <role> to users [from <domain>]". Many participants attempted to express administrators in terms of normal roles and actions; but roles and users are different from resources, and assignment of users to roles cannot be expressed in these terms.

Fundamentally this suggests that users have not understood the difference between RBAC user-role assignment (which are normally done by an administrator) and RBAC role-permission assignments (which are the access control part of the policy).

This is a likely explanation for the observation that participants made this mistake even though the example text gave two clear examples of user-role assignment sentences. This disparity suggests that the example text is not followed closely. However, the observation discussed in section 5.2.4 suggests otherwise. Although there was only a single occurrence early in the trials, the difference between the two cases is striking. It suggests that an example text, no matter how well-written, is unlikely to solve all of the users' conceptual difficulties.

5.4.2 Deny-all-access-except

Another point at which the requirements of PERMIS RBAC diverge from the intuitive expectations of non-specialist users is that, in PERMIS RBAC, the inbuilt default permission is effectively "deny-all-except"; exclusions do not, therefore, need to be explicitly stated, unless it is to reduce the scope of a permission already granted (Brostoff et al's [5] *policy paradigm problem*).

To investigate this, we had taken care to include some exclusions in the scenarios:

Owners *cannot change* any data but can read it all

However, in contrast with Brostoff et al.'s [5] findings using the GUI, none of our participants attempted to express this using an "exclude" clause; a few asked about it, verbally during the trials, but only for clarification, rather than as a conceptual difficulty.

This very positive result suggests that natural language has the potential to overcome conceptual problems which are intractable using more traditional interfaces.

5.5 Analysis

At the end of this subsection we revisit the usability needs which led us to explore the potential of controlled natural language. Before doing so, however, we draw a wider lesson from the results which relate to basic questions in HCI.

5.5.1 Abstraction: What Do They Need to Know?

One of the basic aims of using controlled natural language is that users should be able to specify policies by following examples, with a short learning curve and minimal other guidance [20]. Our investigation asks to what extent this has been achieved in the specific case of the specification of access control policies.

We started from the belief that our target users are "experts" in the access control requirements of the resources under their control. In this research, we found that evaluation participants can follow the PERMIS Policy Editor "dialect" of controlled natural language without difficulty. However, we also found that they needed to know something about the pre-defined "building blocks", while at the same time we have been concerned that they should not have to understand the underlying complexities of the RBAC ontology.

In sections 5.2.4 and 5.4.1, we provided two contrasting examples: in one, participants followed their own partial understanding based on previous experience, while in the other a participant followed the text example in a way which turned out not to be helpful.

The point here is that contents, as well as the phrasing, of the information provided to users are crucial. The example content should naturally lead resource owners to express policies in keeping with the underlying access control model. Yet it should always be possible for a resource owner to express a policy by adapting lines from the example text. Seen in this way, the writing of the example text is a question of *abstraction*. We agree with Witten & Tygar [24] that computer security management, like more conventional programming, is a process of manipulation of abstract rules, and consequently alien and unintuitive for non-programmers. However, abstraction is also the means by which complexity is made manageable [11]. In preparing the example text, the writer/designer is deciding to selectively hide or reveal specific complexities of the underlying system.

5.5.2 Revisiting the Problem

We are now in a position to revisit the usability issues which we identified in section 2.5. Recall that we were concerned with risks in access control specification arising from uncertainty about "*what needs to be done*" [5]. We summarise these as *misconceptions*: security policies are difficult to understand; and *complexities*: the size of the system being controlled makes it difficult to ensure completeness. The user also needs to understand *how to do* what needs to be done. The interface therefore needs to guide a user to produce policies which are accurate, complete, and do not contain security vulnerabilities, and do so this in a way which is intuitive or is available for the user to discover from examples.

In terms of *misconceptions*, our evidence suggests that controlled natural language can reduce the risk and bypass some of the problems found with the GUI interface. In some aspects, such as excluding all permissions by default and being clear about the distinction between subjects ("end users") and targets (resources), it seems reasonable to believe that the logic of the user is intuitively closer to the model when expressing policies in natural language than when using the GUI. On the other hand, not all of the elements of Brostoff et al's [5] *policy components problem* and *policy paradigm problems* are overcome: the distinction between assignments of users to roles and assignment of permissions to roles is still not intuitively understood. Nor is the difference between classes of objects and instances of objects.

A second class of error arises not from misconceptions but from simple "slips" or lapses [8]. In section 5.2.2, we describe the common problem of participants forgetting to pre-specify policy objects, or of being unaware of the need to do so. Yet, as we noted above, this "problem" is also a powerful means to overcome simple errors; it is immediately clear to a user that a mistake has been made. With better feedback, the small problem would be easily overcome, while a larger risk, that of accidentally mis-specifying a policy, is avoided.

In terms of *knowing how to do* it, we feel that the times and numbers of "tries", both overall and for the individual task elements, suggest that controlled natural language allows users to specify accurate and complete policies easily and in a reasonable time. The syntax of the controlled language corresponds well to users' "natural" way of thinking [19] at least in the most basic elements of access control policies: who can perform which actions on which resources. However, there were some common problems which, although ostensibly simple "mistakes", may reflect underlying conceptual difficulties.

6. CONCLUSIONS

Based on the results of the study, constrained natural language is a promising avenue to enable resource owners in Grid computing to express their access control needs more easily. Overall, it provides a usable interface, but some conceptual and usability issues remain. We do not believe that these are insurmountable. We conclude with some pointers, drawn from the results presented here, for ways in which future work can address these issues, and some features which remain to be implemented in our controlled language parser.

6.1 Informing the User

In our analysis (section 5.5.1), we find that designing an interface which supports the user in the cognitive tasks of specifying access control policies, without requiring specialist expertise, is at root a question of abstraction, of ensuring that the user knows enough to avoid mistakes, without becoming overcome by the complexities of the system [24]. In these concluding paragraphs, we move beyond "what does the user need to know", to consider "how does the user know what they need to know?".

The first point of reference which gives guidance to the user is the example text (Figure 1). By design, this text makes no reference to any particular access control context so as to be generally applicable. An obvious response would be to change the design to one in which the example text varies in context. However, this would add complexities of its own and possibly be more confusing for a user. A possible solution would give multiple "typical" examples for different application areas, displayed in tabs; the user would choose the most appropriate example.

The basic problem is to enable the user to understand what is happening when they specify access control polices; if there are rules that fail to parse, the user needs to be able to understand why. Dourish & Button [11] have pointed out that, unlike the real world, computational abstractions are not available to be "*pushed and prodded*" and explored; a static text, by its nature unresponsive and unchanging, exemplifies this problem, but our observations suggest a solution.

In a more dynamic way than the example text, the language parser could provide specific *feedback* to a user. The diagnostic log as currently implemented is only likely to be of interest to a developer. For users, it functions more like a progress bar informing them that processing is still in progress; it provides reassurance that the NLP is not hanging or crashing. However, this log also provides the basis for more useable feedback; several participants drew a comparison with compilers, which provide a comprehensive error report for the benefit of the programmer. It would be helpful if the log could switch between developer mode and user mode. Feedback provided appropriately in this way would immediately overcome the more basic problems, such as that of missing prepositions (section 5.2.3).

6.1.1 Return to the Virtuous Circle

Feedback is not, however, limited to diagnostic output from the language parser. This returns us to the *virtuous circle* of policy specification. We started from the premise that natural language *output* enables the user to check that the machine's understanding of a policy matches with what is intended [8]. With the implementation of controlled natural language input, the virtuous circle is complete.

The diagnostic messages in the natural language output are a key part of helping the user to understand; but the natural language should not be seen as separate from the other interfaces of the PERMIS Policy Editor. During the evaluation, we observed that the existing GUI interface, which we had considered to be a separate part of the PERMIS Editor, was used by participants to understand which parts of their policies had been successfully specified and which had failed. In future work, we plan to link the GUI more closely with the natural language editor, so that modifications made in the GUI are reflected in the natural language text, just as natural language text is already reflected in the GUI. Real-life users also have the availability of the PERMIS Policy Tester, although this did not form part of our evaluations. Thus, as they came to understand the rationality not only of the natural language parser, but of the PERMIS access control system as a whole, participants made use of the interplay between each of the various interfaces to the PERMIS Editor.

The *virtuous circle*, then, can be re-conceived to extend to the PERMIS system as a whole. The specification of policies, like programming, is an iterative process, in which the user is informed by an assemblage of interfaces, working together to ensure accurate and easy expression of authorization policies.

6.2 Remaining issues

6.2.1 Other Access Control

PERMIS is specifically concerned with authorization, but must work alongside other access control mechanisms, such as firewalls and database-level security.

In addition, there are Grid access control requirements that are difficult to implement using RBAC. From our interviews, we noted fine grained-ness and flexibility as core requirements. To support real-world requirements, the variant of RBAC defined by PERMIS supports not only the ANSI standard RBAC features of role hierarchies and dynamic and static separation of duties[3], but also includes enhancements to allow policies to specify other constraints (such as time of day), the use of non-role attributes such as Level of Assurance[4] and delegation of authority. However, at the current time, only role hierarchies, non-role attributes, conditions, and action parameters are expressible in the constrained natural language.

6.2.2 Unique Names in Grid: LDAP

A pre-condition of Grid authorization, and of Grid security as a whole, is that users are uniquely identified, at least within the scope of a Grid [16]. In PERMIS, as elsewhere, this is typically implemented by having items in the policy referred to by Lightweight Directory Access Protocol (LDAP)[5] distinguished names (DNs).

Brostoff et al [5] found that, while the need for unique names is intuitively understood by target users, they are usually not able to correctly specify LDAP DNs; nor should they have to, since the use of LDAP implies that there is a repository which can be searched or browsed for entries.

The GUI part of the PERMIS Editor provides the ability to connect and browse in an LDAP repository. Currently, the controlled language interface does not have any LDAP support, since LDAP is not a part of natural language; users have to browse LDAP via the GUI after they have finished inputting their natural language policy. The natural language output issues warnings that a unique name cannot be found, prompting them to do this. Implementing direct support for LDAP in the language interface will require changes which could also increase the basic usability of the Editor; for example, drop-down menus or hyper-links from which a user could browse an LDAP directory.

[3] ANSI INCITS 359-2004 available at: http://webstore.ansi.org/

[4] NIST Electronic Assurance Guideline 800-63 http://csrc.nist.gov/publications/nistpubs/800-63/SP800-63V1_0_2.pdf

[5] http://tools.ietf.org/html/rfc4510

6.2.3 PERMIS Natural Language in Use

How will people use it? From remarks made by participants during the evaluations and from our interviews and focus groups, we suggest that in real life, people will maintain “scripts” which can be loaded into the natural language interface and amended as needed. If this is correct, then our natural language interface might prove more supportive of maintainability and scalability in Grid access control than the existing GUI interfaces.

7. REFERENCES

[1] Adams, A. and Sasse, M. A. 1999 Users Are Not The Enemy. Communications of the ACM 42,12 (December, 1999), 41-46

[2] Adams, A. and Sasse, M. A. 2001 Privacy in Multimedia Communications : Protecting Users, not Just Data. In: People and Computers XV - Interaction without frontiers. Joint Proceedings of HCI 2001 and ICM 2001 (Lille, France, September, 2001), Springer, Berlin, 49-64

[3] Barman, S., Writing Information Security Policies. New Riders, Indianapolis, IN, USA, 2001

[4] Brodie, C. A., Karat, C.-M., and Karat, J. 2006 An Empirical Study of Natural Language Parsing of Privacy Policy Rules Using the SPARCLE Policy Workbench. In: Proceedings of Symposium On Usable Privacy and Security (SOUPS) (Pittsburgh, PA, USA, July, 2006)

[5] Brostoff, S., Sasse, M. A., Chadwick, D., Cunningham, J., Mbanaso, U., and Otenko, O. 2005 "R-what?" Development of a Role-Based Access Control (RBAC) Policy-Writing Tool for e-Scientists. Software - Practice and Experience 35,9 (2005), 835-856

[6] Cao, X. and Iverson, L. 2006 Intentional Access Management: Making Access Control Usable for End-Users. In: Symposium On Usable Privacy and Security (Pittsburgh, PA, USA, July, 2006), 20-31

[7] Chadwick, D. and Otenko, O. 2002 RBAC Policies in XML for X.509 Based Privilege Management. In: Security in the Information Society: Visions and Perspectives: IFIP TC11 17th International Conference on Information Security (SEC2002) (Cairo, Egypt, May, 2002), Kluwer Academic Publishers, 39-53

[8] Chadwick, D. and Sasse, M. A. 2006 The Virtuous Circle of Expressing Authorisation Policies. In: Proceedings of Second Semantic Web Policy Workshop (SWPW'06) (Athens, GA, USA, November, 2006)

[9] Chadwick, D., Zhao, G., Otenko, O., Laborde, R., Su, L., and Nguyen, T. A. A. 2008 PERMIS: a modular authorization infrastructure. Concurrency and Computation: Practice and Experience Forthcoming (2008)

[10] Charmaz, K., Constructing Grounded Theory: A Practical Guide Through Qualitative Analysis. SAGE Publications, London, UK; Thousand Oaks, CA, USA; New Delhi, India, 2006

[11] Dourish, P. and Button, G. 1998 On "Technomethodology": Foundational Relationships between Ethnomethodology and System Design. Human-Computer Interaction 13,4 (1998), 395-432

[12] Fléchais, I., Mascolo, C., and Sasse, M. A. 2007 Integrating security and usability into the requirements and design process. International Journal of Security and Digital Forensics 1,1 (2007), 12-26

[13] Funk, A., Tablan, V., Bontcheva, K., Cunningham, H., Davis, B., and Handschuh, S. 2007 CLOnE: Controlled Language for Ontology Editing. In: Proceedings of 6th International Semantic Web Conference (ISWC) (Busan, Korea, November, 2007)

[14] Gollmann, D., Computer Security. John Wiley & Sons Ltd., Chichester, UK, 1999

[15] Gollmann, D. 2000 New paradigms - old paradigms? Future Generations Computer Systems 16,4 (2000), 343-349

[16] Humphrey, M. and Thompson, M. R. 2002 Security Implications of Typical Grid Computing Usage Scenarios. Cluster Computing 5,3 (2002), 257-264

[17] Karat, C.-M., Karat, J., Brodie, C., and Feng, J. 2006 Evaluating Interfaces for Privacy Policy Rule Authoring. In: Proceeding of CHI 2006 (Montréal, QC, Canada, April, 2006), ACM

[18] Nielsen, J. Ten Usability Heuristics http://www.useit.com/papers/heuristic/heuristic_list.html

[19] Pane, J. F., Ratanamahatana, C. A., and Myers, B. A. 2001 Studying the language and structure in non-programmers' solutions to programming problems. International Journal of Human-Computer Studies 54,2 (2001), 237-264

[20] Pulman, S. G. 1996 Controlled Language for Knowledge Representation. In: CLAW96: Proceedings of the First International Workshop on Controlled Language Applications (Leuven, Belgium, March, 1996), 233-242

[21] Rode, J., Rosson, M. B., and Pérez-Quiñones, M. A. 2004 End-users' Mental Models of Concepts Critical to Web Application Development. In: IEEE Symposium on Visual Languages and Human Centric Computing (VLHCC'04) (Washington, DC, USA, September, 2004), IEEE Computer Society, 215-222

[22] Saltzer, J. H. and Schroeder, M. D. 1975 The Protection of Information in Computer Systems. Proceedings of the IEEE 63,9 (1975), 1278-1308

[23] Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C. E. 1996 Role-Based Access Control Models. IEEE Computer 29,2 (1996), 38-47

[24] Whitten, A. and Tygar, J. D. 1999 Why Johnny Can't Encrypt. In: Proceedings of the 8th USENIX Security Symposium (Washington, DC, USA, August, 1999), 169-184

[25] Yee, K.-P. 2002 User Interaction Design for Secure Systems. In: Proceedings of 4th International Conference on Information and Communication Security (Singapore, December, 2002), Springer

[26] Zurko, M. E., Simon, R., and Sanfilippo, T. 1999 A User-Centered, Modular Authorization Service Built on an RBAC Foundation. In: IEEE Symposium on Security and Privacy (Oakland, CA, USA, May, 1999), IEEE, 57-71