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 infrastructure to address these challenges. Previous research has found that resource owners who do not understand the PERMIS RBAC model 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 Grid access control needs as expressed by resource owners, through interviews and focus groups with 45 Grid practitioners. 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 participants were not daunted by the text editor, and understood the syntax easily. However, some strict requirements of the controlled language were problematic. Using controlled 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 [19] promoted the security principle of *psychological acceptability*, so that protection mechanisms are applied routinely by their target users. Security which is not usable is likely to lead to dangerous errors [22] and circumventions [1], and ultimately 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 in authorization policies. Security and privacy policies have traditionally been the responsibility of specialists, but the user community is broadening. These disparate groups of users must be enabled to express policies accurately and completely, since security problems may have a highly negative impact [14]. Moreover, security and privacy is rarely the user's main goal. We believe that these observations are equally relevant to the specific area of security in Grid computing on which this paper focuses. 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 [7] offers a basis for achieving usable access control. In essence, PERMIS is an integrated, Role-Based Access Control (RBAC) [21] infrastructure which provides all the necessary support for resource owners to manage authorization policies, and for these policies to be implemented in web services and Grid 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 [6]. To some extent, this need can be met by matching the language of the Editor to that of the target users [15]; earlier work successfully enhanced the usability of the GUI using these principles [4].

The new PERMIS user interface presented in this paper takes a more fundamental approach: it uses *controlled natural language* to reduce the "*distance*" [16] 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*" [15], not by incorporating their language into an interface which still reflects the underlying computer logic, but by enabling resource owners to express policies in their own natural ways of thinking.

In an earlier paper in this project, Chadwick & Sasse [6] asserted that enabling the use of controlled natural language expression of security policies should greatly reduce the scope for *misconceptions* and mistakes in policy specification. 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.

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 enhanced our 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 [11]. 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 target user community for our work was Grid computing; we address problems which have been found in the specification of Grid authorization policies [4]. 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 - 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 [9]. 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 expressing them correctly in an authorization system [11].

We agree with Witten & Tygar [22], whose research into the usability of a public key implementation has relevance to user specification of access control, that computer security management, like more conventional programming, is a process of manipulation of abstract rules, and consequently alien and unintuitive for non-programmers.

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.

2.2 Authorization Reaches Out

In practical implementation, these emerging paradigms can 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 [19].

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. Thus, 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 access control 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, end users, as resource owners, are indeed responsible for setting the access controls. Moreover, as Yee [23] and others have observed, setting and maintaining security controls is rarely the user's primary concern. 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 is all the more important in the growing area of Grid computing. Here, the systems protected, and the applications running on them, are heterogeneous, and may 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 conditions leads to a correspondingly dynamic and complex security configuration.

Moreover, because the computers in a Grid are spread across administrative domains - different organisations and organisational units - and possibly even across jurisdictions or national boundaries, a resource owner will usually not grant access directly to an individual known to him or her. Conversely, with the use of "super schedulers" or resource brokers, a person requesting use of Grid resources might not know in advance the particular set of hosts on which the request will be actioned [12].

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 [16]. More recently, Rode, Rosson, and Quiñones [18] 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; in particular, they found that non-programmer webmasters can usually devise a simple permission 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-*ness* as a way to decrease this "distance" [16]. On the other hand, the experience with early attempts to do so, such as COBOL [20], shows that simply adopting natural-language-*like* syntax does not necessarily lead to naturalness in itself. Pulman [17] suggested that *controlled* natural language, a dialect of English, might be a way to support experts in some field to express their expertise in a way which could be translated into a computer-readable form.

Thus, natural-*ness* is not necessarily best achieved by a full natural language [16]; *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 [10].

The use of specification languages has been used elsewhere in end-user specification of security and/or privacy policies. For example, SPARCLE is designed for natural expression of privacy policies [3]. Adage includes a formal logical language alongside a GUI for expressing RBAC policies [24].

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 users. Our use of controlled natural language is part of our research into ways to overcome these difficulties.

2.5 Challenges in RBAC Policy Specification

Access control in PERMIS is based on a 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 more maintainable and more scalable [5].

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 applications. This is one of its strong points, but without careful interface design this could place a correspondingly greater onus on the resource owner.

However, it cannot be assumed that resource owners are previously familiar with RBAC - especially if Grid use is to extend beyond specialist research. Such resource owners may have a partial understanding of "*what needs to be done*" [4] to implement access control. In particular, Brostoff et al. [4] identified two classes of problem among such target users, the "*policy components*" and "*policy paradigms*" problems.

By "*policy components problem*", they mean a misconception 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*". Policies built using such a mental model are likely to be inefficient rather than insecure, since PERMIS RBAC policies exclude by default all permissions which are not explicitly granted (the "*deny all, except*" model). 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. With increased complexity of the policy, there is a consequent likelihood of omissions, ambiguities or inconsistencies [6]. There can also be mistakes which follow from a basic mis-understanding of the underlying security model, as Brostoff et al. identified [4]. Alongside these new classes of "programming" error, there remains the possibility of simple "slips", lapses or spelling errors, to be eliminated.

For the reasons we discussed above, we believe that controlled natural language has potential for overcoming 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 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 [6]. 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 controlled natural language, rather than being the only means of expressing access control policies. The user can choose either, or switch between them, 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 [5]. The XML is then transformed (using an XSL stylesheet) *back* to the user as (fairly) 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 errors and warning messages. 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.

3.2 Grid Security in the Wild

The process of developing a controlled natural language input for the PERMIS Editor began by interviewing 45 Grid practitioners across the range of application areas: physics, chemistry, 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 and transcribed for analysis in terms of actors, actions, resources, and security needs. 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 controlled 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 present particular challenges, because real-world situations are complex and changeable. Grid computing has varied and sometimes conflicting 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. Even where data is not confidential, integrity is important, especially if 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 commercial value in electronic form, even if the raw data is public. Conversely, the availability of electronic images of rare documents may remove restrictions imposed by the physical vulnerability of the originals.

3.2.2 R-what? Implications for Ontology design

In the absence of easily specified security policies which fit their needs, resource owners may adopt simpler, all-or-nothing policies. Indeed, the evidence of the interviews reinforces the widespread finding that authorization is given a low priority by many resource owners, except in high-security applications. 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, 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 the 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 input of the basic features of the RBAC model. An access control policy in controlled natural language can be thus be transformed into our ontology design, from the ontology into XML, and re-presented back to the user as a diagnostic display, in natural language or another form. However, the parser does not yet include the full functionality of the PERMIS RBAC specification.

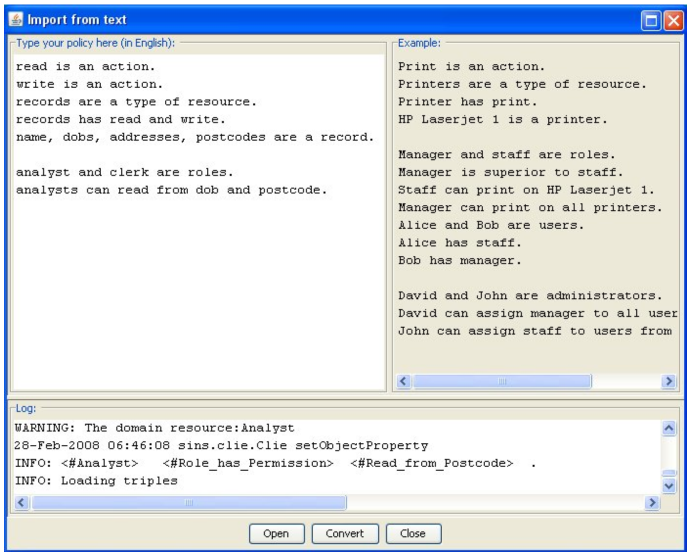
The controlled natural language interface provides a simple layout. On the left-hand side, the user types sentences, each of which represents a rule in the controlled language policy. These sentences do not have to correspond to the order of the final computer-readable policy, but are in any order which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

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

The space for entry of the controlled 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 (Error! Reference source not found.). The right-hand side of the window shows an example of a policy in controlled natural language, 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. Our example text is similar to the Natural Language with Guide described by Karat et al. [13], but shows a complete simple policy by example rather than a guide – thus, users match the example elements to their required policy.

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



```
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

their formal expression. From the implementation point of view, natural language is ambiguous and complex, and consequently very hard to process 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 [6], while still being tailored to the specific requirements of the implementation. In contrast with freer natural language parsers such as that used in SPARCLE [3], accuracy of parsing is not really an issue, since the user is required to keep within the syntax of the controlled language.

Our controlled natural language processor uses 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). CLOnE is built on ten syntactic rules [10], which we have extended slightly. Thus its small rule set and few reserved words avoid the ambiguities of full natural language as well as the compromises of early natural-like programming languages such as COBOL [20].

Yet the constraints of the language also raise the possibility that users' authorization requirements cannot be expressed within its limited syntax. Moreover, not all of the features of PERMIS are currently supported, and in real use it is likely that more needs would be identified.

In effect, specifying an authorization policy is very similar to defining an ontology. On the one hand, this has advantages, because it allows us to start from a well-defined ontology design and to adapt an existing controlled language for our purpose. On the other hand, the critical point for usability is that the ontology should remain transparent to the user.

3.4.1 The Controlled Natural Language Interface

Some powerful usability features of CLOnE have been carried over into our controlled 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 controlled natural language, we have extended CLOnE in three 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 clerk 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.*".

The third of these changes reflects the important of trust in the access control policies; however, this was not a focus of this research and did not form part of the observations.

3.4.2 Classes and Instances: New Entities from Old

Our parser provides a useful grouping feature, which allows users to refer concisely to properties which apply to the whole group.

A feature native to CLOnE, is that entities can be created as a "type of" some already existing entity - that is, as a sub-class of a class in the ontology.

We have extended this feature so that, when an entity class is created, a special pseudo-instance is automatically created, called "*all_<class>*" (eg. "all_Printer"). This can be used later to refer to every object of that type (every instance of the class or subclass). For example,

> Printers are a type of resource.
>
> Floor6_Color_printer is a printer.
>
> Managers can print on all printers.

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 controlled 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. Users do not know, and should not have to know, about the predefined ontology; but they should still be able to specify policies by following the example text.

It is important to emphasise that users are not expected to know, or to need to know, anything about the underlying ontology or rulesets; we discuss these here only to clarify the connection between controlled 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.

The aim is that users are able to specify polices with the need only to learn a few simple rules and follow the example text.

4. EVALUATION

The interviews conducted in the first phase informed the requirements for the ontology design, which is the basis for a controlled natural language interface. 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), and use them to construct policies?
3. Can target users avoid misconceptions in the RBAC/PERMIS model when using the PERMIS controlled 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 in controlled natural language. 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. In real life, as users have to do more than comply with tasks as they are prompted by a scenario [22], but at this stage we were interested specifically in their use of controlled natural language. Within the limitations of the scenario, a "correct" policy is one in which everything which should be specified, is specified, and all permissions which should be granted, are granted.

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 controlled 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 field and some understanding of Grid computing), senior research management (Principal Investigators) and administrators (such as departmental administrators or information systems staff). Although they have good computer skills, they are not computer security specialists.

All 17 participants were from our target group of users; all were highly computer-literate and working in a variety of computer-related areas. All of the participants were fluent in written and spoken English, although not all had English as a first language. Reflecting our aim of investigating security policy authoring by users who are not Grid or security specialists, these are target users even though not all of the participants had specific e-Science or Grid experience.

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 were library computer professionals (web and database administrators, project managers).

4.1.2 Conducting the observations

Although resource owners, if they are not security specialists, may not have a detailed understanding of RBAC, we believe it is realistic to expect that they 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, we prepared a short (1 page) description of the basic RBAC concepts. This was read to them by the experimenter in order to overcome different abilities in grasping written information and to allow the experimenter to check understanding at key points.

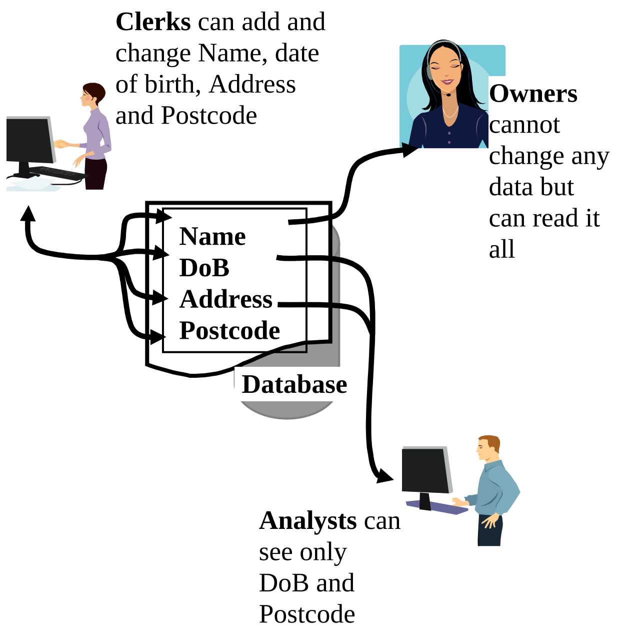

Clerks can add and change Name, date of birth, Address and Postcode
Owners cannot change any data but can read it all
Name
DoB
Address
Postcode
Database
Analysts can see only DoB and Postcode

Figure 2: General scenario 1 (diagrammatic form)

Each participant was then given, in printed form, the first scenario presented in two formats: a 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 mainly follow 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.

The first, simpler, scenario contained three roles, three resources with three possible actions on them, and one administrator. The second scenario was 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: explicit 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]. Participants were allowed to ask questions and the observer could intervene at his discretion. 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 inform the results.

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".

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

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, a simple 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: 14 (of 17) participants grasped the basic concept of expressing policies in controlled language without difficulty; the other 3 initially attempted to use more uncontrolled language, but were nevertheless able to grasp the controlled syntax with some intervention by the observer. 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, 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 controlled 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 from the "handover" of the session, the delay time before the participant started to type on the text editing space was an average of 3:35 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 47 minutes and 10 tries, although 7 required more than trivial help to do so; 11 of 17 needed less than 30 minutes. By "non-trivial help", we mean that the observer intervened to overcome some of the conceptual problems which we discuss later in this section. In other cases, users were able to identify and overcome these problems unaided or only requiring help in, for example, a mis-spelling. Overall mean times to complete the scenario (with some observer intervention in 6 cases) were 30:01 minutes in 5.41 tries. Excluding two outliers, mean times for completion of the first scenario were 24:27 minutes in 4.47 tries.

We hope that the overall times will fall as target users learn the requirements of our controlled natural language. There is some evidence to support this. Of the 9 participants who proceeded to the second, slightly more advanced, scenario, the mean time was 15:43 minutes in 2.67 tries, considerably 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

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.

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 dynamically; 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 administrators; the aim is to prevent mistakes introduced by typing errors. For example:

> Clerk can read from *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. In our evaluations, this kind of error occurred most frequently with *actions*: 7 users failed to specify at least some of the actions to which they later referred.

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.

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. Currently, the appropriate design response is to guide users to follow the text accurately.

This is not a fundamental issue in the gap between users' intentions and their expressions of them in language, but illustrates the difficulties of making the language truly natural, and problems if it falls short of this aim.

5.3 Building Policies from Ontology Elements

As we reported above, the basic entities were well understood. 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 exposing explicitly the design of the ontology.

5.3.1 Understanding the policy "building blocks"

Participants should not have to know about the ontology and the "building blocks" of an access control policy; they should be able to specify policies intuitively from the example text. 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.

This is the "grouping" feature of our controlled language, described in section 3.4.2, a powerful tool for users but also presents users with the possibility of unproductive choices. Misunderstanding this, 5 participants defined elements as *types* of resource, that is, as an ontology subclass, rather than as *instances*:

> Postcode is a type of resource.

instead of, for example:

> Field is a type of resource.
>
> Postcode is a field.

The problem with the first of these declarations is that unless the user specifies actual instances of a subclass into which the "*all_<class>*" pseudo-instance is then expanded, no actual resources are created in the final policy. But it is unlikely that a non-specialist user, with no knowledge of the ontology, would deduce this. Worse, policies expressed in this way can parse and convert apparently correctly, but do not, in fact, contain any resource instances and hence also no permissions. Some participants, using the GUI, discovered that their subsequent policies did not contain any resources when viewed, but were unable to explain this without intervention.

All but one of these participants subsequently expressed resources in a way which instantiates, either as above - as instances of a subclass which they defined, such as *Field* - or, more simply - but contrary to the example text - as instances of *Resource*:

Name, Address, Dob and Postcode are Resources.

Fundamentally, users should not be expected to 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 subclass; *AA1 1BQ* could be an instance of this subclass. Users should be guided to use whichever form is most appropriate, using the more powerful grouping form if appropriate.

5.3.2 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 – permissions apply to the instances of resources and actions on them, not to 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. The original example text was more complete, and in a sense more correct, but also potentially confusing for users.

5.4 Overcoming Misconceptions

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

5.4.1 Where users are from

Notably, the participants were able to express the *target domain/subject domain* distinction which was a source of misconception for Brostoff et al [4]**,** the first aspect of their *policy components problem*. Indeed, in the controlled natural language this distinction is largely avoided, so that it becomes intuitively obvious that DepartmentA is the domain from which requests originate, the *subject* domain, in the example text:

John can assign staff to users from DepartmentA.

This is despite some problems in practical application. The first version of the example text omitted this crucial sentence; an experimental shortcoming which, nevertheless, generated useful insights into the ways in which participants are able to use the example text. Secondly, the syntax at this point is strict, and spaces in particular cause problems for the parser. Finally, the choice of "staff" as a role name is potentially confusing.

Of the 5 participants who completed the scenario with the amended example text, 3 were able to express assignments to departments correctly and the remaining 2 with a very little assistance, suggesting that this is intuitively followed.

From this evidence and from the interactions in the recordings, combining "*where users are from*" with the authority of administrators in one sentence creates a positive support for users' understanding. However, concerning another point of misconception identified by Brostoff et al. [4], the function of domain administrators and the separation of roles from end-users, the evidence suggests that this remains problematic, as we describe in the following section.

5.4.2 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>]". Five participants attempted to express administrators in terms of normal roles and actions, such as "*add*" and "*remove*"; but roles and their assignment are separate from permissions to act on resources, and assignment of users to roles cannot be expressed in these terms. Instead, the special verb "assign" is used for this.

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

This is a likely explanation for the observation that participants made this mistake even though the example text gave clear examples of user-role assignment sentences.

5.4.3 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 [4] *policy paradigm problem*).

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

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

Brostoff et al.'s [4] predicted that novice users would have a mental model similar to "*explicitly grant and deny access*". However, only two of our participants attempted to express this using an explicit deny; a few others asked about it, verbally during the trials, for clarification. This suggests that controlled natural language has the potential to overcome conceptual problems.

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 Human-Computer Interactions.

5.5.1 What Do They Need to Know? How can they know it?

One of the basic aims of using controlled natural language is that users should be able to specify policies by following example rules with minimal other guidance [17]. The example text is therefore crucial in this respect.

In sections 5.3.2 and 5.4.2 we provided two contrasting examples of participants making use of the example text. In the first, a participant followed the text example in a way which turned out not to be helpful, while in the second, participants failed to follow the example accurately, instead superimposing their own partial understanding. Similarly, participants who forgot to put prepositions between verbs and resource names were failing to follow the example text - a warning given during the introduction to the experiment also reminded them of this requirement.

The example content should naturally lead resource owners to express policies in keeping with the underlying access control model. The example must be as clear and simple as possible; yet it should always be possible for a resource owner to express a policy by adapting lines from the example text. The ways in which participants do, or do not, make use of the example text is indicative of the importance of linguistic cueing [8] for users. Even though users knew that they had to express their intentions subject to a controlled language, their ordinary natural language use led to these simple syntactic mistakes.

5.5.2 Thinking about language

The issues which we have identified are not all simple usability problems that could have been identified by heuristic analysis. The issue with "*type of resource*" rather than "*resource*" is subtle, and illustrates the importance of user testing.

Any language processor has to interpret the user's intentions. As we explained, in a different situation, there might be a need to exploit the grouping features of the language with, for example, Postcode as a subclass. The challenge for us is to enable users to understand the choices open to them, while helping them to avoid mis-specifying policies which do not function as intended.

This complex interplay between users' own expectations and the features of an interface suggests that controlled natural language alone is not sufficient to solve the problems we have identified. Rather, the existing interfaces should work with controlled natural language, to allow users to disambiguate their intentions and to provide better and more immediate feedback.

5.5.3 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*" [4] and *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 in some aspects, such as excluding all permissions by default, it seems that the logic of the user is intuitively closer to the model when expressing policies in controlled natural language than when using the GUI. On the other hand, not all of the elements of Brostoff et al's [4] *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. There is a new source of misconception in the distinction between subclasses of objects and instances of objects.

A second class of error arises not from misconceptions but from simple "slips" or lapses [6]. 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 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 policies easily and in a reasonable time. However, there were some common problems which, although ostensibly simple "mistakes", may reflect underlying conceptual difficulties.

6. CONCLUSIONS

We started from the belief that our target users are "experts" in the access control requirements of the resources under their control. The question is whether they are able to express this expertise using a quite general controlled natural language.

We found that evaluation participants were able to follow the PERMIS Policy Editor "dialect" of controlled natural language. However, we also found that, while they intuitively understood the pre-defined "building blocks", they sometimes had problems in knowing how to construct policies from these "building blocks". Our implementation is promising to the extent that it decouples intention from the underlying model; problems arise when the controlled language implementation fails to match natural language or the users' intuitive understanding.

From our experimental method, we also learnt a lot about the importance of the example, and of the careful wording of the scenario. This is most evident in the confusion of subclasses and instances. The example text shows actions on an object – a named printer - which is naturally understood as one object (one instance), and groups this into the class, "Printer", with its associated set, "All Printers". Our scenario, in contrast, referred to a set of objects - fields in a database - as though they are a single object. This basic scenario was not based on Grid or e-science; its abstractions might have been more difficult to express than, say, access to physical objects or to a file. However, while this might be considered a flaw, it revealed a problem which, we believe, is always present: objects can always be grouped into sets in ways which may, or may not, be supportive of the users' intentions.

We conclude with some suggestions, drawn from the results presented here, for ways in which future work can address the problems we have identified, and some features which remain to be implemented in our controlled language parser.

6.1 Informing the User

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 (Error! Reference source not found.). By design, this text makes no reference to any particular access control context so as to be generally applicable. One 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. The question of how to bridge the subtle distinction between classes and instances for non-specialists raises difficult usability issues, and we are working on design ideas for how this might be achieved. The special *all_<class>* (section 3.4.2) instance is a partial solution - actions can be given to all instances of a resource class. However, the user still needs to specify the resource instances into which this special instance is later expanded. We see a more dynamic and more supportive language interface, working in conjunction with the GUI, prompting users to disambiguate, and to fill gaps in their specifications.

The diagnostic log, currently only likely to be of interest to a developer, 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; in user mode, feedback could be immediate, rather than waiting for the user to “Convert” the entire policy specification.

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 [6]. 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 this should not be seen as separate from the other interfaces of the PERMIS Policy Editor. During the evaluation, we observed the ways in which participants made use of the existing GUI interface 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 controlled natural language editor, so that modifications made in the GUI are reflected in the language text, just as language text input 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.

The *virtuous circle*, then, can be re-conceived as the various interfaces to the PERMIS Editor working together to help users to understand the rationality not only of the language parser, but of the PERMIS access control 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, combining to ensure accurate and easy expression of authorization policies.

6.2 Remaining issues

6.2.1 Other Access Control

This early implementation does not yet support all of the features available or which are being developed in PERMIS RBAC. It does support role hierarchies and parameters to actions, but not, currently, ANSI standard RBAC features such as separation of duties, other conditions such as time of day, authorization decisions based on non-role attributes such as Level of Assurance[3], obligations, and dynamic delegation of authority.

A final requirement of PERMIS policies is that resources are tied to actions, so that only permitted actions can be performed on target resources (although an action can, alternatively, be allocated to “all targets”). In the GUI, this is labeled “*Resources’ Functions*”; in PERMIS terms, this is the Action Policy. This is the rather non-natural “*Printer has print*” form in controlled natural language. However, although this form is parsed, it is not currently implemented in the policies which are produced, and was ignored in our scenarios, although some participants, following the example text, specified it. It is nevertheless indicative, again, of the use made of the example text, and of the acceptability of this rather non-natural language, constrained by the features of the underlying platform.

6.2.2 Unique Names in Grid: LDAP

A pre-condition of Grid authorization, and of Grid security as a whole, is that users have a Grid-wide uniquely identity [12]. In PERMIS, as elsewhere, this is typically implemented by having items in the policy referred to by Lightweight Directory Access Protocol (LDAP)[4] Distinguished Names (DNs).

Brostoff et al [4] 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; users have to browse LDAP via the GUI after they have finished inputting their controlled natural language policy. 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] NIST Electronic Assurance Guideline 800-63 http://csrc.nist.gov/publications/nistpubs/800-63/SP800-63V1_0_2.pdf

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

6.2.3 PERMIS Controlled 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 controlled natural language interface and amended as needed. If this is correct, then this might overcome issues of scalability in our controlled natural language interface and also in working with the existing GUI interfaces.

Although we were constrained by the use of a particular controlled natural language platform, placing an ontology, rather than an XML expression of policy, at the centre, potentially enables a more flexible design with new ways of interrogating an policy. The issues we have identified in this early version of the language parser provide pointers for improvements but also suggest more general conceptual issues for future research.

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, Germany, 49-64

[3] 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)

[4] 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

[5] 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, Dordrecht, Germany, 39-53

[6] 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)

[7] 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)

[8] Clark, L. and Sasse, M. A. 1997. Conceptual design reconsidered - the case of the internet session directory tool. In: People and Computers XII: Proceedings of HCI'97 (Bristol, UK, August, 1997), Springer, Berlin, 67-84

[9] 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

[10] 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)

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

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

[13] 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

[14] Karat, J., Karat, C.-M., and Brodie, C. Human-Computer Interaction Viewed from the Intersection of Privacy, Security, and Trust. In The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications Sears, Andrew and Jacko, Julie A (Eds.) CRC Press, Boca Raton, FL, USA, 639-658

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

[16] 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

[17] 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

[18] 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, 215-222

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

[20] Sammet, J. E. The early history of COBOL. In History of Programming Languages Wexelblat, Richard L (Ed.) New York, NY, USA, ACM, 199-143

[21] 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

[22] 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

[23] 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, Berlin, Germany

[24] 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