Controlled natural language policies*

Juri Luca De Coi
Forschungszentrum L3S
Appelstr. 9a
Hannover, Germany
decoi@L3S.de

Peter Fankhauser
Forschungszentrum L3S
Appelstr. 9a
Hannover, Germany
fankhauser@L3S.de

Tobias Kuhn
University of Zurich
Binzmühlestrasse 14
Zurich, Switzerland
tkuhn@ifi.uzh.ch

Wolfgang Nejdl
Forschungszentrum L3S
Appelstr. 9a
Hannover, Germany
nejdl@L3S.de

Daniel Olmedilla
Telefónica R&D
C/ Emilio Vargas, 6
Madrid, Spain
danieloc@tid.es

ABSTRACT

Policy languages allow users to define in a flexible way under which conditions their data can be shared and with whom. Nonetheless policy languages are not widely used yet: one reason for this is that they are too complex to be easily exploited by common users in a profitable way. This paper describes an approach that uses (controlled) natural language in order to express policies, therefore allowing non-computer experts to easily specify, refine and understand policies.

1. INTRODUCTION AND MOTIVATION

Recent experiences with Facebook's *beacon* service[1] and Virgin's use of Flickr pictures[2] have shown that the widespread adoption of applications (like Web 2.0 applications and social softwares) which allow users to share data did not come along with an equally widespread use of technologies (such as policy languages) which allow users to flexibly define which data can be shared with whom. One reason for this is that current policy languages are too complex to be easily exploited by common users in a profitable way, and that typically require a policy writer to be a computer expert[3].

This paper introduces an approach to express policies by means of controlled natural languages. In particular, we allow users to specify policies in the ACE controlled natural language [7], which are automatically translated to the PROTUNE policy language. The use of controlled natural languages to express policies drastically improves the usability of policy languages and we believe that it has the potential of fostering that broader adoption of policy languages which nowadays is still lacking.

The remainder of this paper is organized as follows: Section 2 presents the concept of "policy" and introduces the PROTUNE policy language. Section 3 presents the concept of "controlled natural language" and introduces the controlled natural language ACE. Section 4 introduces the mapping we defined between ACE sentences and PROTUNE policies. Section 5 presents a predictive editor which guides the user step by step toward defining PROTUNE policies by means of ACE sentences. We conclude in Section 6.

2. THE CONCEPT OF "POLICY"

According to [11]'s well-known definition, policies are "rules governing the choices in the behavior of a system", i.e., statements which describe which decision the system must take or which actions it must perform according to specific circumstances. *Policy languages* are special-purpose programming languages which allow to specify policies, whereas *policy engines* are software components able to enforce policies expressed in some policy language.

Policies are encountered in many situations of our daily life: the following example is an extract of a return policy of an on-line shop[4].

> Any item for return must be received back in its original shipped condition and original packing. The item must be without damage or use and in a suitable condition for resale. All original packaging should accompany any returned item.

With the digital era, the specification of policies has emerged in many web-related contexts and software systems. The use of formal policies yields many advantages compared to conventional approaches: formal policies are usually dynamic, declarative, have a well-defined semantics and allow to be reasoned over.

*The authors' efforts were partly funded by the European Commission in the TENCompetence project (IST-2004-02787) (http://www.tencompetence.org).

[1] http://www.washingtonpost.com/wp-dyn/content/article/2007/11/29/AR2007112902503.html?hpid=topnews

[2] http://www.smh.com.au/news/technology/virgin-sued-for-using-teens-photo/2007/09/21/1189881735928.html

[3] "Too often, only the PhD student that designed a policy language or framework can use it effectively", Kent Seamons, Semantic Web Policy Workshop, 2005.

[4] http://www.fancydressking.co.uk/returns-policy/info_5.html

Hyatt Regency Chicago Chicago, Illinois, USA

Policies have been attracting pretty much attention for the last years: many policy languages have been defined and many engines able to enforce policies expressed in such languages have been implemented (cf. [5] for a broad overview of the research field as well as a comparison of the existing solutions).

In this paper we describe our experience with PROTUNE: according to [6, 5], PROTUNE is one of the most complete policy languages to date and has been already applied to a number of scenarios, e.g., learning environments [4], the development of a security layer on top of generic RDF stores [1], and enforcing efficient policy-aware access to semantic metadata stores [8].

3. CONTROLLED NATURAL LANGUAGES

Controlled natural languages (CNLs) are subsets of natural languages that are restricted ("controlled") in a way that reduces or removes the ambiguity of the language (see e.g.,[10]). The main motivation is to improve the human-computer interaction: the users should be able to express statements in a language that is familiar to them. On the other hand, the restrictions of the language enable automatic processing by computers.

The controlled natural languages we are talking about in this paper are completely formal languages that can be mapped automatically and unambiguously to formal representations. Such languages are described by a formal grammar and are usually explained to the users on the basis of construction rules ("which subset of the natural language is covered?") and interpretation rules ("how are the sentences interpreted that would be ambiguous in the natural language?").

In this paper we describe our experience with ACE: Attempto Controlled English (ACE)[5] [7] is a mature controlled natural language (concretely a controlled subset of English) that has been developed and constantly extended during the last 13 years. Initially designed as a specification language, the focus has shifted towards knowledge representation and especially towards applications for the Semantic Web.

4. MAPPING ACE TO PROTUNE

This section outlines the principles we tried to stick to when designing the ACE→PROTUNE mapping. We point the reader to [3] for a thorough description thereof.

Defining a mapping between ACE sentences and PROTUNE policies involves two main tasks: (i) characterizing the subset of ACE that expresses PROTUNE policies, and (ii) mapping this subset to PROTUNE. In both steps we strove toward translating linguistic constructs of ACE into PROTUNE linguistic constructs which have a comparable semantics. Further design principles we tried to stick to when developing the ACE→PROTUNE mapping are outlined in the following list.

Coverage The mapping should cover PROTUNE as much as possible, i.e., one must be able to define as many PROTUNE policies as possible through ACE sentences: it is indeed the case that all PROTUNE policies not containing metarules nor `in`-predicates (cf. [2]) can be expressed by means of suitable ACE sentences. On the other hand not all possible ACE sentences can be translated into PROTUNE policies. The reason for this is that many of the linguistic constructs of ACE do not have an immediate correspondence in PROTUNE

Unambiguity The mapping must be unambiguous, in the sense that for each covered ACE sentence there must be only one PROTUNE policy it is translated into (i.e., the input ACE sentence must univocally determine the PROTUNE policy it is translated into)

Syntax-based As described in Section 3, only syntactic information is typically exploited by interpreters of controlled natural languages. They typically do not exploit contextual information, which can help humans to disambiguate ambiguous sentences, but resort to a set of built-in disambiguation rules. Our mapping must also only use syntactic information

[5] `http://attempto.ifi.uzh.ch/site/`

5. USABILITY ISSUES

The mapping introduced in Section 4 has been implemented as a command-line tool: the user is expected to provide an ACE sentence as input and the tool will deliver as output a PROTUNE policy. Whenever a non well-formed sentence is input, an error message containing debug information as well as suggestions to fix the problem occurred is shown (e.g., *Every ACE text must end with . or ? or !.*).

Although meaningful error messages can help by driving users toward well-formed input sentences, the approach provided by such command-line tool suffers from a big drawback in terms of usability: even if the user fixes some error in her sentence according to the suggestions contained in the error message, there is no guarantee that the corrected sentence will be well-formed, since it may still contain (different) errors. For this reason the process of inputing a sentence can result in a (potentially long) sequence of steps, each giving rise to new errors which need to be fixed.

W.r.t. usability, a much better approach is to constrain users to only create well-formed input sentences: Section 5.1 describes a possible way to enforce this constraint.

5.1 A predictive authoring tool

Even though CNLs are much easier to use than other computer languages, they still require a minimum learning process for their restrictions and interpretation rules. In order to ease and speed up this learning process, we suggest to use predictive editors [9], i.e., editors that are aware of the grammar of the used CNL and that can guide the user step by step through the creation of a sentence. Such an editor forces the user to continue the sentence in a way that corresponds to the respective grammar. Therefore, a complete sentence is always syntactically correct and no error messages are needed.

Figure 1 is a screenshot of the predictive editor that we developed. The numbered components are explained below.

(1) is a read-only text area that shows the beginning of a sentence. This fragment has been entered by a user and it has been accepted by the editor as a correct sentence beginning. Thus, there is at least one possible completion that leads to a correct sentence. The button `Delete` can be used to go back, whereas pressing the `Clear` button resets the content of the text area.

The text field (2) can be used for entering the next words of the sentence. If they are a correct continuation of the

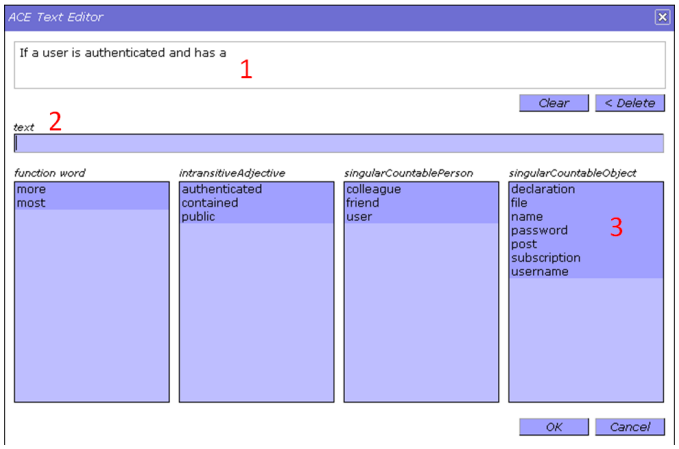

Figure 1: A predictive authoring tool

sentence then they are moved to the text field (1) as soon as the `RETURN` key is hit.

Clicking on the entries of the menu boxes (3) is an alternative way to construct a sentence. There is a menu box for each word class that is allowed at the current position. In this case, only function words, intransitive adjectives, or singular countable nouns representing persons or objects are allowed. The menu box for verbs, for example, is not shown because verbs are not allowed at this position.

5.2 Editor's features

As described in Section 4 not all features of ACE are used to express PROTUNE policies. For this reason the set of all ACE policies (i.e., ACE sentences expressing PROTUNE policies) is a proper subset of the set of all ACE sentences. Our editor (shown in Figure 1) currently supports a proper subset of ACE policies, which we called *ProACE*. This set is sound and possesses a remarkable property which we call *half-coverage*.

Soundness Each sentence belonging to ProACE can be translated to a PROTUNE rule. This property ensures that it is never the case that an ACE sentence created by means of the editor cannot be translated into a PROTUNE policy

Half-coverage For each ACE sentence which (i) does not belong to ProACE; and (ii) could be in principle translated to a PROTUNE policy P; there is a(n ACE) sentence belonging to ProACE which can be translated to P. This property ensures that, if there is at all a way to express a PROTUNE policy in ACE, then there is a way to express the same policy in ProACE. In other words, this property guarantees that, although ProACE is a proper subset of ACE, it does not lower ACE's expressiveness w.r.t. the capability of expressing PROTUNE policies

6. CONCLUSIONS

This paper presented an approach to policy specification based on the use of controlled natural languages. Controlled natural languages provide major advantages over usual formal languages in terms of usability. Usability can be further improved by providing the user with suitable editing tools.

In this paper we applied our approach to the PROTUNE policy language and the controlled natural language ACE. Moreover we presented an editor which guides the user step by step toward defining policies by means of controlled natural sentences.

Although the need for data protection is strongly perceived, today's technologies like policy languages, which allow to flexibly define which data can be shared with whom, are far from widespread. We believe that the use of an interface based on controlled natural languages has the potential of fostering a broader adoption of policy languages and we believe that our work is a first step in this direction.

7. REFERENCES

[1] F. Abel, J. L. De Coi, N. Henze, A. W. Koesling, D. Krause, and D. Olmedilla. Enabling advanced and context-dependent access control in rdf stores. In *ISWC/ASWC*, pages 1–14, 2007.

[2] P. A. Bonatti and D. Olmedilla. Driving and monitoring provisional trust negotiation with metapolicies. In *POLICY*, pages 14–23, 2005.

[3] J. L. De Coi. Mapping natural language to formal policies: an ace→protune mapping. Technical report, Forschungszentrum L3S, July 2008.

[4] J. L. De Coi, P. Kärger, A. W. Koesling, and D. Olmedilla. Control your elearning environment: Exploiting policies in an open infrastructure for lifelong learning. *TLT*, 1(1):88–102, 2008.

[5] J. L. De Coi and D. Olmedilla. A review of trust management, security and privacy policy languages. In *International Conference on Security and Cryptography (SECRYPT 2008)*, 2008.

[6] C. Duma, A. Herzog, and N. Shahmehri. Privacy in the semantic web: What policy languages have to offer. In *IEEE POLICY'07*, 2007.

[7] N. E. Fuchs, K. Kaljurand, and T. Kuhn. Attempto Controlled English for Knowledge Representation. In C. Baroglio, P. A. Bonatti, J. Małuszyński, M. Marchiori, A. Polleres, and S. Schaffert, editors, *Reasoning Web, 4th International Summer School 2008, Venice, Italy, September 7–11, 2008, Tutorial Lectures*, number 5224 in Lecture Notes in Computer Science, pages 104–124. Springer, 2008.

[8] E. Ioannou, J. L. De Coi, A. W. Koesling, D. Olmedilla, and W. Nejdl. Access control for sharing semantic data across desktops. In *PEAS*, 2007.

[9] T. Kuhn and R. Schwitter. Writing Support for Controlled Natural Languages. In *Proceedings of the Australasian Language Technology Workshop (ALTA 2008)*, 2008.

[10] R. Schwitter, K. Kaljurand, A. Cregan, C. Dolbear, and G. Hart. A Comparison of three Controlled Natural Languages for OWL 1.1. In *4th OWL Experiences and Directions Workshop (OWLED 2008 DC)*, Washington, 1–2 April 2008.

[11] M. Sloman. Policy driven management for distributed systems. *J. Network Syst. Manage.*, 2(4), 1994.