# A Semantic Policy Management Environment for End-Users and Its Empirical Study

Anna V. Zhdanova, Joachim Zeiß, Antitza Dantcheva, Rene Gabner, and Sandford Bessler

**Abstract.** Policy rules are often written in organizations by a team of people in different roles and technical backgrounds. While user-generated content and community-driven ontologies become common practices in the semantic environments, machine-processable user-generated policies have been underexplored, and tool support for such policy acquisition is practically non-existent. We defined the concept and developed a tool for policy acquisition from the end users, grounded on Semantic Web technologies. We describe a policy management environment (PME) for the Semantic Web and show its added value compared to existing policy-related developments. In particular, we detail a part of the PME, the policy acquisition tool that enables non-expert users to create and modify semantic policy rules. An empirical study has been conducted with 10 users, who were new to the semantic policy acquisition concept and the developed tool. The main task for the users was to model policies of two different scenarios using previously unknown to them. Overall, the users successfully modeled policies employing the tool, with minor deviations between their performance and feedback. Observation-based, quantitative and qualitative feedback on the concept and the implementation of the end-user policy acquisition tool is presented.

## 1 Introduction

Community-driven services and portals unifying physical and virtual realities, such as 43things.com, SecondLife, YouTube, LinkedIn and Facebook, or the Web 2.0 developments, are currently at their popularity peak attracting millions of users. The

Anna V. Zhdanova, Joachim Zeiß, Rene Gabner, and Sandford Bessler
Telecommunications Research Center Vienna (ftw.), Donau-City Strasse 1,
A-1220 Vienna, Austria
e-mail: {zhdanova,zeiss,gabner,bessler}@ftw.at

Antitza Dantcheva
Eurecom, BP 193, F-06904 Sophia Antipolis, France
e-mail: antitza.dantcheva@eurecom.fr

S. Schaffert et al. (Eds.): Networked Knowledge - Networked Media, SCI 221, pp. 249–267.
springerlink.com 

existing portals with their community and personal data management environments, while collecting and attempting to manage large amounts of user-generated content, are still highly limited in providing the functionality assisting adequate management and sharing of the submitted data. In most cases, the users still cannot specify the provisioning conditions of the generated content or services (i.e., policies with whom and for what they want to share), as well as set up automatic execution of arbitrary actions provided that the certain conditions are met (e.g., notifications about appearance of specific information, products, services or user groups). Ability to define and employ policies would lead to efficient personal information management, decrease amounts of electronic spam, and increase revenues for targeted provisioning of content and services.

Semantic Web and social software technologies have proved to be a success in resolving the knowledge acquisition bottleneck. Approaches such as Semantic Wikis [7], [10] enable acquisition of large quantities of arbitrary ontology instance data. Community-driven ontology management [11], [12] shows feasibility of acquisition of ontology classes, properties and mappings from the end user communities. Meanwhile on the large-scale light-weight and tag-based social Web, *user-generated policies* have not yet gained a broad usage. The latter is largely due to the complexity of this problem w.r.t. the user perspective [5] and a lack of practices and tools for policy acquisition from the non-expert users.

The main contributions of the presented work are:

- Definition of a user-driven policy management environment for open, sharable infrastructures such as for Web or mobile services,
- A concept and a tool for policy acquisition from the end users, grounded on Semantic Web technologies.
- An empirical study conducted to test the approach and the tool. The study shows that end users modeled the policies employing the tool successfully and are inclined to use similar tools in the future.
- Observation-based, quantitative and qualitative feedback on the end user policy acquisition concept and the implementation is presented. Requirements towards the design of an improved policy acquisition tool are drawn.

The paper is structured as follows. In Section 2, we describe our approach of a policy management for the Social Semantic Web. In Section 3, we describe the potential research applicability and the related work. The implementation of the policy acquisition tool is presented in Section 4. The user study and tool evaluation settings are described in Section 5. In Section 6, the results of the study are presented, and lessons for the future construction of policy acquisition tools are drawn. Section 7 concludes the paper.

## 2 Semantic Policy Management

The following paragraphs describe the basic components of our architecture. The architecture is strongly related to conventional ontology and policy management services [2], [5], [9], but is enriched with end-user generated policy acquisition and

advanced policy communication. The basic model is that of an open system in which policy rules can be shared, adapted to individual needs and enriched with facts and instance combinations.

A **Policy Storage and Query** component is provided to efficiently store and query parts of policy data and metadata by providing indexing, searching and query facilities for ontologies. In addition to conventional policy management services and practices [2], [5], [9], we propose to enrich the existing search and query components with community-generated policy information. This would improve their performance and make the search, reasoning and consistency checking features mature and more attractive to use.

As the users of the environment are generally not bound to a single community or application, they must be able to publish personal and community-related policies in a multi-accessible way. The current focus in semantic policy storage and querying is thus maintaining distributed repositories with functionalities for aggregation, decomposition and discovery of information in simple ways.

A **Policy Editing** component is introduced for creating and maintaining policies and instance data. The front-end, a user-friendly interface, helps users to easily add and modify policy-like rules on the basis of existing imported ontology classes and properties shared among several users and communities, policies and instances. The back-end consists of a storage and query system. A Policy Editor enables sharable editing for multiple users and tight integration with semantic publishing, delivery and visualization components, allowing the involved parties to observe the evolution of policy settings. These requirements are due to the elevated degree of flexibility required by community-oriented environments as the Social Semantic Web and its members to freely evolve schemata, policies and to influence community processes.

A **Policy Versioning** component is introduced to maintain different versions of policy definitions, as communities, content and relationships change over time. The user should be able to easily adapt policies to new scenarios and communities without losing previous definitions. Earlier versions can be reused for definitions of new policies. Also users could experiment with more restricting policy definitions and roll back to previous versions wherever practical. A Policy Versioning component interacts with existing versioning systems like svn [3] to provide a versioning service to the user. Semantic metadata describes the necessary versioning information inside the policy definition itself.

A **Policy User Profile and Personalization** component is responsible for the users' access to the environment and it connects the policies with the user profiles. At a more advanced level, the component helps to share and communicate policies across the users' profiles, apply policies dependent on the user profiles and recommend policies based on the user profiles. In particular, access and trust policies can be implemented taking into consideration community and social networking information provided by the users [7].

Our ***overall*** ontology-based **policy management** approach features: ***user-driven policy construction***, meaning that the system extensively assists the users to model the policies correctly (e.g., proactive suggestion of the ontology items that can be combined in a policy, consistency checking for the modelled policy solutions);

***policy semantic representation and sharing across communities,*** essential for the further extension for the rules layer of the Semantic Web; ontology import and ***policy creation on the basis of shared ontologies,*** the user is free to input any ontologies he/she likes and define policies on them.

Thus, ontology-based and community-oriented policy management is an advance over a conventional policy management. The advantages are gained by introducing an infrastructure that enables the communities to manage their policies.

## 3 Research Applicability and Related Work

In this section, we discuss the applicability of the approach and related work in the field of (semantic) policy editing.

### *3.1 Applicability of Policy Acquisition Tools*

In a distributed environment, such as the Internet, there is a need to set policies for sharing user information and for providing access to services on the Web. However, the ways to model and operate with Semantic policies are currently very limited, and there are *little or no approaches for policy acquisition from the end users*. Meanwhile enabling the end users to define and share policies is crucial for widespread, acceptance and the growth of the rule-based Semantic Web.

The types of users of a Policy Acquisition Tool (PAT) include (but are not restricted to) the following:

- Individual users who have one or several profiles and have to manage them on several systems (related to single-sign-on systems);
- Owners of web services who want to sell or offer their functionality to others and need to specify the conditions under which the service can be used;
- Users who manage the physical reality or link physical and virtual worlds. For instance, such user activities include setting policies on forwarding phone calls from the user's phone to his/her mobile phone while he/she is on vacations, or sending an SMS to a remotely-located mother if her baby wakes up and starts to cry, employing integration with the sensor technology.

The customers of a PAT would be companies or institutions:

- providing single-sign-on applications;
- providing identity management and security systems (e.g., for the users who want to specify different user groups on an instant messenger and show their location information only to some of these groups);
- developing aggregation solutions for systems with similar functionalities for users to have one profile and a possibility to set various policies for various systems (such as Trillian for instant messaging);

- providing (semantic) web service publication space and (semantic) web service search engines for web service owners to annotate their services with specific service features (e.g., conditions for execution);
- providing community sites where people add/create content (such as Flickr for pictures), so that the users set policies on how and by whom their content can be accessed and used;
- providing systems for the management of physical environments such as semi-automatic policy-based assistance in a hospital on observing patients, notifying nurses, etc.

### *3.2 Related Work*

In current software products, policy editing often can be performed in a simple manner, in particular, via checkboxes and scroll-down forms. Well-known examples of such policy management include a policy editing interface for handling files in Microsoft Word. For the user convenience multiple templates are offered for selection of the rule type that the user may want to edit, e.g., in Microsoft Outlook. Also, in the contemporary applications, web-portals and online shops, the users are often asked to commit to agreements or copyrights written in a natural language by clicking an "accept" button. Such agreements or copyright statements in particular may contain policies on the e-mail addresses and personal data sharing, the users' preferences, etc.

Techniques from the following *research* fields are relevant for the user-oriented policy acquisition:

- Policies on the Semantic Web in general: state of the art in this area and the new trends (e.g., automated trust negotiation) are overviewed by Bonatti et al. [2]. As a particular effort, Attempto Project[1] has developed tool support for transferring statements (possibly, user rules) specified in controlled natural language (English) to the OWL format [10];
- Policies have been applied to web services [13], and "The Web Service Policy Framework"[2] is an example of an industry-led effort in this area;
- Ontologies for defining policies: a number of works are driven in this area, for instance, an ontology for defining business rules in OWL [12];
- Knowledge acquisition methods for ontology construction, including knowledge acquisition principles in ontology editors, community portals [16];
- Editing of policies: an editor developed by Karat et al. [8] is one of the advanced works most strongly related to our work and therefore we go into detail when explaining the differences to our work. The editor has three variations of policy editing for the end user: (i) Unguided and (ii) Guided Natural Language interfaces, and (iii) Structured List method, i.e., the interface allowing composition

[1] Attempto Project: http://attempto.ifi.unizh.ch/site/description/index.html

[2] The Web-Service Policy Framework: http://www-106.ibm.com/developerworks/library/ws-polfram/

of policies out of pre-existing items, via web form-based selection. One of the test-based observations was that "Structured List methods helped the users create more complete rules for all element categories except Conditions as compared to the Unguided NL method".

In our work, the ontology-based policy engine is structurally similar to the Structured List method, i.e., major part of the policy specifications are created from pre-existing components (in our case, ontology and instance data items). Using ontology technologies brings an added value to the conventional policy editing technologies due to the following factors:

1. End users define the policies easier and faster than with other methods (including both natural language and Structured List methods) due to the fact that the users are familiar with ontologies employed for their profile or context information;
2. The composed set of rules is even more complete than the most complete set obtained by now (with the Structured List method);
3. While the Structured List method was shown to be the most preferred method by users, the ontology-based method would become the first preferred method.

There are also *tools* enabling the users to edit the policies:

- PERMIS[3] has similar policy editing functionalities as addressed here (e.g., for personal data protection), however, the tool does not include semantic policies, there the policies are specified in XML which hinders referencing or reusing items of already existing ontologies. In addition, the tool is restricted for scenarios specific for settings of certain existing platforms (namely, Apache Web server, Globus Toolkit, Shibboleth, .Net, Python interfaces);
- WebSphere Policy Editor[4] is an Eclipse plug-in tool for generating, creating, and editing cache policies (as in PERMIS, based on XML) for the dynamic cache service of WebSphere® Application Server;
- P3PWiz[5] is an online commercial tool by Net-Dynamics allowing website owners to design P3P[6] compliant policies via graphical interfaces. Mainly predefined selection forms are used, which are tightly compliant to the fixed P3P specification. Other similar online services are P3PEdit[7] and P3PWriter[8]. IBM's P3P Policy Editor[9] is also a similar tool which is downloadable as a JAR file.

The policy constructions supported in these tools are restricted to certain domains and do not allow inclusion of arbitrary ontology-based vocabularies. As new domains and ontologies always appear and evolve in community user-driven systems, inability to support construction of policies in dynamic semantic environments is a severe bottleneck of the aforementioned tools.

[3] PERMIS: http://sec.cs.kent.ac.uk/permis/
[4] WebSphere Policy Editor: http://www.alphaworks.ibm.com/tech/cachepolicyeditor
[5] P3PWiz: http://www.p3pwiz.com
[6] P3P, the Platform for Privacy Preferences: http://www.w3.org/P3P/
[7] P3PEdit: http://p3pedit.com
[8] P3PWriter: http://www.p3pwriter.com
[9] IBM P3P Policy Editor: http://www.alphaworks.ibm.com/tech/p3peditor

## 4 Policy Acquisition Tool

In this section, we describe the implemented tool for a policy acquisition from end users: its general overview, functionality and user interfaces.

### *4.1 Tool Overview*

The implemented policy acquisition infrastructure is designed as a component for a community Semantic Web portal, providing policy management facilities to the community members and managers. The infrastructure is built as a Web-based application using JSON technology[10] [4] and exploiting Python version of Euler [6] for manipulating ontology schemata, instance data and policies in a N3 format [1]. A policy is modeled as a rule in a typical form of one or more *conditions* followed by one or more *conclusions*.

The architecture of the community-driven policy acquisition infrastructure is shown in Figure 1. The policy acquisition tool (PAT) is facilitated by another major block, the policy engine (PE). A PAT server is a component interacting with the end-user over a GUI, and the policy engine is a component responsible for the "logical" side of the system, accomplishing integration of external and internal information, reasoning and rule production. The PAT server is the active component addressing the policy engine with requests whenever the user loads a policy, selects the policy building blocks or saves a policy.

The policy engine (PE) is a stateless request-/response-based server that deals with any kind of requests expressed in N3 [1]. The policy engine has associated a *Decision Space*, a set of files containing N3 triplets as well as rule objects, i.e., parsed N3 statements, kept in memory. The files contain persistent semantic data like ontology definitions, instance data and rules. Volatile semantic data relevant for the current policy request are added to the N3 objects in memory. The *Request Processor* is the part of the PE that extracts data from the request (out of a SIP message, a http GET/POST message or a SMS) and inserts it into the decision space. The policy engine may also extract data from a user profile, user context such as location, or policy data via an additional context interface. The *Reasoner*, the heart of PE, is a N3 rule engine that is invoked with the receipt of a request and uses all semantic data made available in the decision space as reasoning input. The reasoner is based on the python implementation of Euler (backward-chaining, enhanced with Euler path detection).

The tool applications comprise usage and population of domain-dependent and domain-independent ontologies, and service support for the portals' data and metadata. The prototype is using N3 notation due to its simplicity and efficiency in representation of the rules, and a straightforward integration with other semantic languages such as RDF/S and OWL. Technically, the rules written in N3 syntax are compact enough to be effectively executed and managed even on devices with limited computational and storage capacities, such as mobile phones. The overall

[10] JSON: http://www.json.org

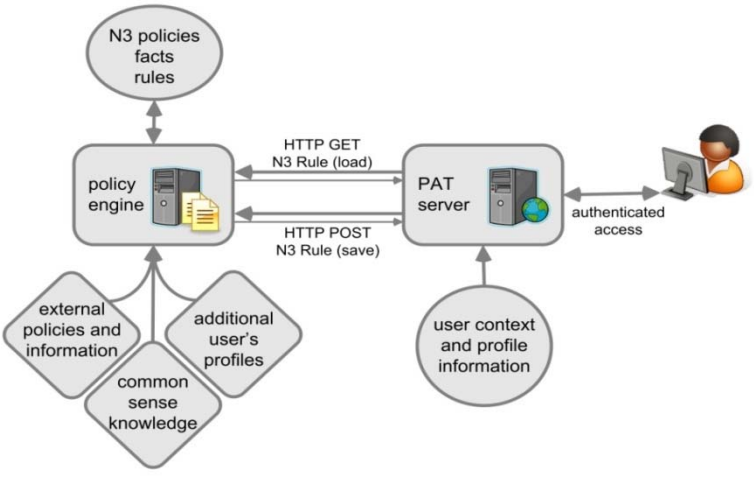


**Fig. 1** Policy management infrastructure

size of the ontologies employed in the presented user studies comprised 94 ontology items (including classes, properties, instances) for the Eshop case study and 127 items for the Etiquette case study.

## *4.2 Tool Functionality*

Below we list the functions and features of PAT. Currently most of the listed functions and features have been implemented and the others are being implemented.

**PAT functions** are as follows:

- *Viewing a policy/user rule:* With PAT, the user can view all the constructed policies, possibly divided into groups of rules;
- *Dynamic user interface generation:* The user interface is generated directly from the ontologies and the instance data that are imported by PAT. The ontologies and data can be provided by the end user(s) or deduced by the policy engine based on defined business logic. For example in Table 1, PAT recommends the user to choose between the objects that are compatible with the subject "Maria";
- *Modifying a policy/user rule:* By loading a specific rule from the policy engine's decision space, it is possible to modify existing rules, i.e., either rules generated by the current user or by other users;
- *Assisted fill out:* User profiles and context ontologies are employed to assist the user in filling out the policy items when modeling a rule. In the shown experiments here, PAT interface offers only combinable data according to the context ontologies;
- *Deleting a policy/user rule:* Alike to modifying a rule, it is also possible to delete the currently edited rule from the policy engine space. A deletion immediately effects queries from other clients;

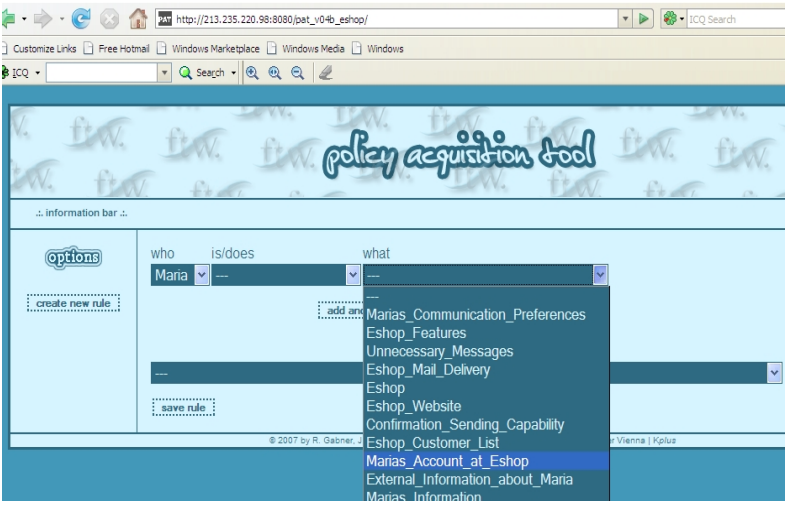


**Fig. 2** Interface for policy acquisition: policy construction

- *Saving a policy/user rule:* Once the user has successfully finalized a rule, the rule can be saved directly to the policy engine's decision space (as a N3 language notation text file). Figure 3 shows an example of the finalized rule;
- *Human readable format rule tagging:* Naming a policy or user rule for further reference and search enables the users to retrieve the rules employing the rules' human-readable names or tags. All PAT functions (viewing, modifying, deleting and saving) benefit from this human readable format.

An example of a policy that is valid for an Eshop case study on online shopping (see Appendix) and is applicable to a hypothetical online customer Maria is "We might receive information about you from other sources and add it to our account information". This policy is being designed in PAT's user interface in Table 1 and 3, and its N3-based representation is as follows:

```
Maria a :Customer.
Eshop a :Eshop.
External_Info_about_Maria a :External_Customer_Info.
Marias_Account_at_Eshop a :Eshop_Customer_Account.
{
  Maria :has Marias_Account_at_Eshop.
  Eshop :receives External_Info_about_Maria
} => {
  External_Info_about_Maria
      :is_added_to :Marias_Account_at_Eshop
}
```

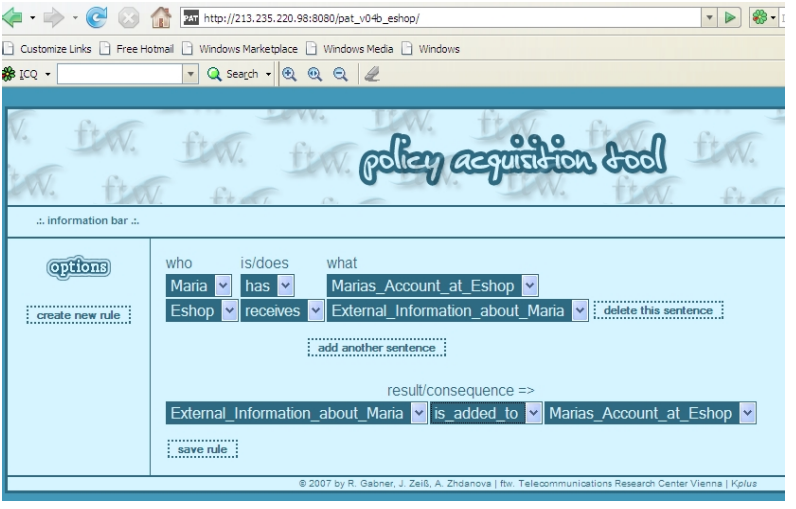


**Fig. 3** Interface for policy acquisition: policy is finalized

As the user in question is already familiar with Eshop and its terms, the selection of the relevant concepts for the rule in the interface becomes easier for him/her.

## 5 User Study

In this section, we explain the goals of the user study, and its set up and procedure.

**Goals of the user study** were as follows:

- To investigate the users' attitudes towards applying a semantic policy editor;
- To identify usability problems of the evaluated PAT interface and to derive suggestions for its improvement;
- To gain a first evidence about the following theses:
  - The editor will contribute to the widespread use of policies on the web and in mobile environments (e.g., the users use the tool well and think that they will use it in the future);
  - Policy acquisition tools are applicable both to private data management and business/company settings;
  - The editor reduces the costs of policy construction and makes the policy design accessible to non-professionals;
  - The editor is helpful in reducing the mistakes in the process of the rule construction;
  - The editor makes the user more aware of policies and encourages the user to observe, construct and manage them.

**User study setup.** The user study contained 10 test persons of different age (between 25 and 35), sex (4 female, 6 male), education and technical experience. To obtain a MOS (mean opinion score) the subjects were asked to rate the criteria on a five grade scale (1-fully disagree, 3-undecided, 5-fully agree) with half-grades possible (e.g., 2.5). The working device was a laptop placed on a table. The test took place in the HTI (human telecommunication system interface) lab, simulating a home atmosphere. A test conductor was observing and taking notes during the test.

**User study procedure.** The test with each user lasted for up to one hour and comprised the steps as specified in Table 1. A questionnaire, the log files created automatically during the test and the filled out observation forms were used to analyze the user study.

## 6 Results

In this section we present and discuss the results obtained in the user study. The first subsection discusses findings derived from the observation of the users while they were constructing policies. The second subsection summarizes and discusses the results obtained from questioning the users about the policy acquisition tool.

### *6.1 Observation-Based Results*

In this subsection we present the user study results (obtained via observation of the users) and a discussion. In particular, we explored how people started and proceeded with the construction of a policy, to which extent the resulting policies were complete and correct, how much time the user spent on the construction of a policy, how well the user understood modeling of more complex rules comprising such constructions as multiple conditions and consequences, active vs. passive voice or negation.

All the users without any exception started to model the first policy beginning with the first graphical condition field ("subject" of the statement), see Figure 2. Later on, the users discovered that modeling a rule from any graphical slot was possible: 4 out of 10 users discovered and started using this feature themselves and the remaining 6 users did the same after an indication from a test conductor. The feature of constructing the complex rules (the "Add sentence" button) on the interface drew more attention: 7 out of 10 users noticed it and started to try it out proactively themselves, and only 3 users ignored the feature until the test conductor drew the attention to it.

Another interesting observation was that the users were generally inclined to model at first the consequence of the rule and then the condition(s), which was contradictory to the modeling suggested by the user interface. Six out of 10 users started modeling with representing the consequence prior to the condition: as a rule they modeled the consequence statement in the condition statement, realizing their misplacement afterwards. Three users demonstrated such behavior persistently, i.e., more than once during the test.

**Table 1** Test roadmap

| Step | Details |
|---|---|
| 1. Introduction to the user study | A description of the type of assessment, the opinion scale and the presentation of the policy editor were given in oral form before the beginning of the test. |
| 2. Explanation of the Eshop use case scenario to the user | The first scenario Eshop is detailed in Appendix, Case Study 1. |
| 3. First scenario: Eshop policy modeling tasks done by the user | The user was told that he/she was to act as a manager of an internet shop (Eshop). He/she had to model the online shop's privacy policies in a policy editor for the customers. The customer was named Maria: "Maria regularly shops online, likes special offers and recommendations, but wants to keep her personal profile information under control." The user was to model 5 policies according to that case within the policy management environment. |
| 4. Explanation of the Etiquette use case to the user | The second scenario Etiquette is detailed in Appendix, Case Study 2. |
| 5. Second scenario: Etiquette policy modeling tasks done by the user | Here the user had to implement behaviour rules valid for different situations. The policies modelled would be incorporated in software to help children, foreigners, robots or automated personal assistants to make choices about their behaviour. Again 5 tasks for various situations (e.g., tea party, restaurant) were to be implemented by the user. |
| 6. Overall questionnaire answered by the user | A final questionnaire provides the information about the attitude of users towards the policy editor. |
| 7. Test completion | |

In ca. 90% of the cases policies were modeled correctly by the user, i.e., the resulting rule conveyed the same meaning as the one offered to be modeled. Almost every second "correctly modeled" rule was represented using the same vocabulary and the same level of precision as implied initially with the given ontology. Assuming the community-driven rule development and sharing, incomplete rules (e.g., lacking certain condition or consequence statements) or rules using alternative vocabularies could achieve the same level as the completely and precisely modeled rules after being augmented with the context information and ontology mappings repositories [15]. The ratios for the users' policy modeling success rates are presented in Figure 4.

An *average amount of time* spent on the construction of a policy was 3.5 minutes (Table 1, 3 and 5). Major delays have been caused by a reasoner taking the time to upload the matching statements and the recurring need to re-model certain parts of the rules when they proved to be unfitting. As the users were getting familiar with the tool, the time spent on a construction of a typical policy approached one minute. Technically, the spent time can be reduced by a more scalable implementation.

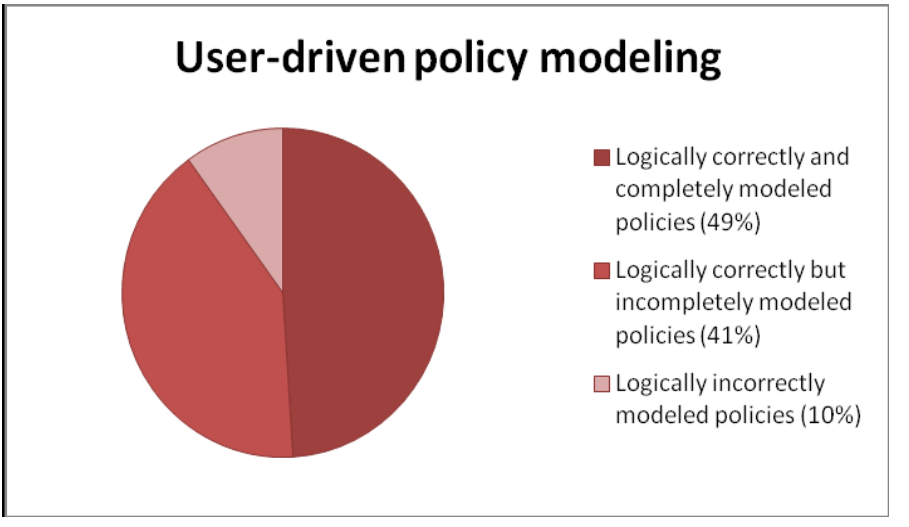


**Fig. 4** Policy modeling success rates obtained in the experiment

Additionally, we observed the users' reaction on various logical constructions: (i) rules containing negation statements, (ii) rules containing statements represented both in active and in passive voices, (iii) rules containing more than one condition statement, and (iv) rules containing more than one conclusion statement.

For the rules containing *negation statements*, the test included not appearing vocabulary to express a statement with a negation and the test leader suggested the user to model the same statement using a positive construction. In this case, 6 out of 10 users modeled the statement positively indicating that the modeled statement is not the same as the provided statement, while the other 4 users modeled the statement without indicating the difference.

A mix between *active and passive voices* in the statement descriptions and models has been explicitly indicated as an obstacle by 2 users out of 10. Though the majority of the users (8) did not raise the issue of an active and passive voice mix, in many cases the users were expecting usage of a certain voice in the statement model. Occasionally incorrect expectations in the voice matter lead to longer rule construction times as the users had to re-model the rule parts that are constructed based on wrong assumptions.

Rules with *more than one condition* or *more than one consequence* were modeled more imprecisely than rules with only one condition or consequence by the user in 5 and 1.5 cases out of 10, accordingly. There, in 3 and 6 cases out of 10, the user still modeled the rule with only one condition or consequence, though giving a note on imprecise modeling. Only in 1 and 2.5 cases out of 10, the users implemented such complex rules logically correctly via finding less trivial tool usages, in particular, representing the semantic of one rule by modeling two separate rules.

## 6.2 Questionnaire-Based Results

The questionnaire consisted of nine questions. Users were asked to answer all the questions. Under every question they could give a grade from 1-5, where 1 meant "fully disagree" and 5 meant "fully agree". Below we present the questions, the feedback and the recommendations provided by the test persons verbally. In Figure 5, the mean scores of the grades for all the questions and all the users are summarized.

*Question 1: Was it easy for you to understand the system?*
Users liked the minimalistic design (that led to fewer distractions) and that learning to use the tool is easy. After about 2 accomplished modelled policies, the users were familiar with the policy editor and could fully concentrate on the more complex tasks and not the system itself. They rated the easiness of use of the system with a MOS of 3.6 (standard deviation of 1.15) that is above average. The difficulties in understanding the system mentioned by users were:

- Active / passive or negatively formulated policies: the policy linguistic description and its ontology vocabulary could mismatch the users' expectations;
- Distinguishing between "condition" and "consequence" of the policy: the users attempted to model a consequence in a place of a condition;
- Two users mentioned that editing the rules starting from the consequence is easier;
- Speed: the loading time was too long, what made users impatient to try further combinations;
- Clarity and visibility: an overview of the selectable conditions and results at each time was desired.

*Recommendations:* Keeping a simple and overview giving design; clear and obvious explanations of entities; clear labeling; improving the speed (e.g., via caching of

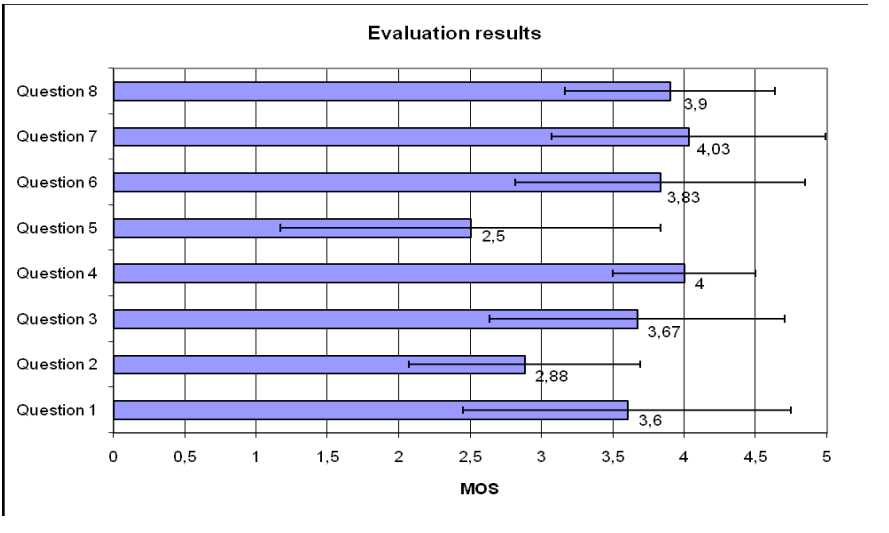


**Fig. 5** Policy modeling success rates obtained in the experiment

information); specialized and customized policy editors, kept simple with well-chosen expressions.

*Question 2: Was it easy to find/select the right terms to express your ideas?*
The obtained MOS was 2.88 (standard deviation of 0.81) for the expressive potential of the policy editor. Here users again pointed out that they would like to be able to change all the fields anytime and that they had difficulties in choosing the direction of implementing ("subject – predicate – object" or "object – predicate – subject"). Also the negation containing policy and synonym search were confusing.
*Recommendations:* flexible system, easy to edit and change yet simple; well-chosen available vocabularies.

*Question 3: Was it easy to combine simple sentences in more complex constructions to express your ideas?*
Users found it easy to combine simple sentences in more complex constructions (MOS=3.67, standard deviation 1.03). The question here appearing is about the logical operation relating to two or more conditions. However, 5 persons left out this part, needing more explanations and time than they were willing to spend;
*Recommendations*: anticipate complex structures and define the connections and relations between the options to choose properly.

*Question 4: Did the system work in such a way as you expected?*
The novelty of policy acquisition tools as a class leads to mostly no exact expectations. One person was expecting to write the policies in words without further options, whereas another thought of a graphical display. Still in general the implemented system matched the test subjects, resulting in a MOS grade 4.00 (standard deviation 0.5) for this question.
*Recommendations:* considering a tree-like design: visualization of conditions and rules as a graph overview.

*Question 5: Do you think that the amount of time you generally have spent on the construction of a policy is adequate?*
The most improvable factor of the policy editor is time consumption. The loading time on the one hand and users' thinking time on the other hand are influencing this aspect, leading to the rating of 2.50 (standard deviation 1.33).
*Recommendations:* More simplicity and technical optimization are desired here.

*Question 6: Can you imagine yourself using the policy editor in the future for managing your personal data?*
With a MOS rating of 3.83 (standard deviation 1.01), the users were willing to use policy editors in future for managing their personal data. After experiencing the system 9 users had positive attitudes towards policy editors, one user could not think of an application. The only concerns were related to privacy, security and transparency of the tool.
*Recommendations:* Privacy and security procedure should be declared.

*Question 7: Can you imagine using the policy editor professionally or privately for defining policies to offer services or sell products?*

Here users were very convinced (highest rating of the study: 4.03; standard deviation 0.96) that they would chose to use the policy editor. The difference to the previous question is that the application did not concern employment of personal data (but offering a service / selling a product). In addition, the users would like to have more options to select, simpler (well-constrained) domains and to have a "testing function" in order to make sure that the modeled policies lead precisely to the result that the user wants.
*Recommendations:* testing function; available overview of all implemented rules.

*Question 8: Do you think that you would use this or a similar system regularly in the future?*
Again a relatively high MOS grade of 3.9 (standard deviation 0.74) was given. The users are eager to use policy acquisition tools on a regular basis in the future. They found the system "helpful".
*Recommendations:* topic-related customization.

*Question 9: What was your overall impression of the policy editor? What are its strengths and weaknesses? Please provide us any kind of feedback in a free form.*
The overall perception was very positive. Users liked the idea of editing policies, the user interface and found the usage easy after a short period of learning how the tool works. Again the problems noted above were pointed out (improveable speed, difficulties with more complex rule structures, "consequence vs. condition" modeling, more intuitive vocabularies. An improved PAT is expected to be received even better.
*Recommendations:* Improved modelling principles, in particular, based on recommendations from further user tests or placing the system online.

## 7 Conclusions and Discussion

We see the following value being added by an ontology-based policy management compared to conventional policy practices:

1. ***Spreading of policies***, freedom in policy distribution and sharing, annotation of the end users' data and services, easyness in reading other people's and organizations' policies; all this would be difficult without the semantic practices.
2. ***Reduction of costs for policy construction***: existing similar policies may be available and easy to reuse elsewhere. For example, most of the internet shops have very similar polices on how to deal with the customer data and they would not need to redefine all the policies from scratch. One could also advance eGovernment visions by provisioning machine readable laws, e.g., on data protection,
3. ***Reduction of the mistakes in the user-generated policy modeling*** as the system's storage, query and reasoning service as well as sharing of policies within communities act as controllers for policy correctness.
4. ***Better awareness of the end users about policies, rules and regulation:*** With the suggested system the policies are easily retrieved and presented to the users.

Policy editors emerge as new assistance tools, allowing users to define rules in various settings. The user study showed that users came along very well with the policy modeling tasks, without special preparations or much prior knowledge of the concept. Overall, the test subjects felt consistently very positive about the introduced policy acquisition tool. In particular, the standard deviation of users' evaluation on the overall tool usage (question 1) is 1.15, which shows that the opinions have had minor differences within the user set. The test subjects were commonly eager to use similar tools for private as well as professional purposes: question 8 related to this criterion has the mark higher than the average and the rather low standard deviation of 0.74. Most of the involved persons appreciated the benefits of the system (in particular, saved costs, policy management by the end consumer and the reduction of the modeling mistakes).

The occurred usability issues define requirements for the next versions of the tool. In particular, the response delay mainly depends on the complexity of requests regarding vocabulary retrieval and validation, a missing caching mechanism of inference results and the reasoning strategy of the policy engine. The currently rather generic yet complex request of the PAT server towards the policy engine should be split in a small and simple set of specific requests to obtain vocabulary and validate usage. Furthermore, it should be investigated to find the optimal reasoning algorithm for different types of policy requests and scenarios. The user interface could be simplified by offering different levels of verbosity for editing policies. Distinct vocabularies or placements of statement slots should be offered for editing condition and conclusion triplets.

Apart from technical and usability issues, the following more socially-oriented questions should be investigated in community-driven policy modeling studies:

- How users share personal data, multiple identities, etc. Initial observations can be drawn from social networking websites (e.g., LinkedIn, Xing, etc.) where users can select whether they share a specific type of information with other users;
- Specifying, accumulating and storing arbitrary policies could result in a "policy Wikipedia" provisioning commonsense knowledge rules of what users find right and appropriate, e.g., "do not drink and drive". Such community effort would also have an anthropological effect in enabling observation of which kind of policies are shared between large communities and which policies are less popular.
- Certain policies vary by countries, cultures and time (e.g., eating any kind of food using hands could have been acceptable in certain countries in the past, but not in the present). This adds to additional technical challenges in policy versioning, matching and comparison.

Finally, we are convinced that policy acquisition from the end users is a highly important functionality for services offered in user-centered open environments, such as in the (Semantic) Web or mobile settings. Also we foresee that implementations of such ontology-based policy acquisition will become essential for any end user-oriented environment involving policies, such as business and private data management tools, ubiquitous environments, eGovernment applications.

**Acknowledgements.** The Telecommunications Research Center Vienna (ftw.) is supported by the Austrian government and the City of Vienna within the competence center program COMET. This work is partially funded by the EU IST projects Magnet Beyond (http://www.ist-magnet.org) and m:Ciudad (http://www.mciudad-fp7.org).

## Appendix: Case Study Descriptions

**Case Study 1: Eshop**
You are a manager of an internet shop (Eshop). You need to model for an Eshop customer the following privacy policies in a policy editor. The customer's name is Maria.

1. Information You Give Us: we store any information you enter on our website or give us in any other way.
2. You can choose not to provide certain information but then you might not be able to take advantage of many of our features.
3. Automatic Information: we store certain types of information whenever you interact with us.
4. E-mail Communications: we often receive a confirmation when you open e-mail from Eshop if your computer supports such capabilities.
5. We also compare our customer list to lists received from other companies in an effort to avoid sending unnecessary messages to our customers.

**Case Study 2: Etiquette**
You need to model the following policies related to basic human behavior or etiquette in a policy editor. The policies modeled by you would be incorporated in software to help children, foreigners, robots or automated personal assistants to make choices about their behavior.

1. Restaurant. What about Doggy bags? There's nothing wrong with taking your leftovers home in a doggy bag, especially since portions are usually more than any human should eat in a single sitting.
2. Tea Party. Since it is a tea party, it's okay to eat with fingers.
3. Phone Conversation. While answering a call, do not scream or use a harsh voice.
4. Phone Conversation. In case of a poor connection or when you are abruptly disconnected, the individual who originated the call is responsible for calling back the other party.
5. Phone Conversation. If you want to leave a voice mail message on the phone, repeat your name and telephone number twice, clearly.

## References

1. Berners-Lee, T., Connolly, D., Kagal, L., Scharf, Y., Hendler, J.: N3Logic: A Logic for the Web. Journal of Theory and Practice of Logic Programming (TPLP) (2007) (Special Issue on Logic Programming and the Web)

2. Bonatti, P.A., Duma, C., Fuchs, N., Nejdl, W., Olmedilla, D., Peer, J., Shahmehri, N.: Semantic web policies - a discussion of requirements and research issues. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 712–724. Springer, Heidelberg (2006)
3. Collins-Sussman, B., Fitzpatrick, B.W., Pilato, C.M.: Version Control with Subversion. O'Reilly, Sebastopol (2004)
4. Crockford, D.: The application/json Media Type for JavaScript Object Notation (JSON). RFC 2647 (July 2006)
5. Davies, J., Fensel, D., van Harmelen, F.: Towards the Semantic Web: Ontology-Driven Knowledge Management. John Wiley & Sons, Chichester (2002)
6. De Roo, J.: Euler proof mechanism (2007), http://www.agfa.com/w3c/euler/
7. Golbeck, J., Parsia, B., Hendler, J.: Trust Networks on the Semantic Web. In: Proceedings of Cooperative Intelligent Agents 2003, Helsinki, Finland (2003)
8. Karat, C.-M., Karat, J., Brodie, C., Feng, J.: Evaluating Interfaces for Privacy Policy Rule Authoring. In: Proceedings of the Conference on Human Factors in Computing Systems (CHI 2006), pp. 83–92 (2006)
9. Kaviani, N., Gasevic, D., Hatala, M., Wagner, G.: Web Rule Languages to Carry Policies. In: Proceedings of Eighth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2007), pp. 188–192 (2007)
10. Kuhn, T.: AceRules: Executing Rules in Controlled Natural Language. In: Marchiori, M., Pan, J.Z., Marie, C.d.S. (eds.) RR 2007. LNCS, vol. 4524, pp. 299–308. Springer, Heidelberg (2007)
11. Riehle, D. (ed.): Proceedings of the 2005 International Symposium on Wikis (WikiSym 2005), San Diego, California, USA, October 16-18 (2005)
12. Sriharee, N., Senivongse, T., Verma, K., Sheth, A.P.: On Using WS-Policy, Ontology, and Rule Reasoning to Discover Web Services. In: Aagesen, F.A., Anutariya, C., Wuwongse, V. (eds.) INTELLCOMM 2004. LNCS, vol. 3283, pp. 246–255. Springer, Heidelberg (2004)
13. Verma, K., Akkiraju, R., Goodwin, R.: Semantic Matching of Web Service Policies. In: Proceedings of the Second International Workshop on Semantic and Dynamic Web Processes (SDWP 2005) (2005)
14. Völkel, M., Krötzsch, M., Vrandecic, D., Haller, H., Studer, R.: Semantic Wikipedia. In: Proceedings of the 15th International conference on World Wide Web, WWW 2006, Edinburgh, Scotland, May 23-26 (2006)
15. Zhdanova, A.V., Shvaiko, P.: Community-Driven Ontology Matching. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 34–49. Springer, Heidelberg (2006)
16. Zhdanova, A.V.: Community-driven Ontology Construction in Social Networking Portals. International Journal on Web Intelligence and Agent Systems 6(1), 93–121 (2008)