Exploiting Policies in an Open Infrastructure for Lifelong Learning

Juri L. De Coi, Philipp Kärger, Arne W. Koesling, and Daniel Olmedilla

L3S Research Center and Leibniz University of Hannover, Hannover, Germany
{decoi,kaerger,koesling,olmedilla}@L3S.de

Abstract. Nowadays, people are in need for continuous learning in order to keep up to date or be upgraded in their job. An infrastructure for lifelong learning requires continuous adaptation to learners needs and must also provide flexible ways for students to use and personalize them. Controlling who can access a document, specifying when a student may be contacted for interactive instant messaging or periodical reminders in order to increase motivation for collaboration are just some examples of typical statements that may be specified by e.g., learners and learning management system administrators. This paper shows how policies can represent a way of expressing these statements and describes the extra benefits of its adoption like flexibility, dynamicity and interoperability.

1 Introduction

Society and current labor market evolves rapidly. Nowadays, a learner is potentially any person in the world, who wants to keep up to date on any specific topic, be it at work or in any other facet of her life. Therefore, there is a growing need for more flexible and cost-effective solutions for learners allowing them to study at different locations (e.g., at home) and at times that are better arranged with their working hours. In addition, learners do not necessarily work isolated but may collaborate with or contact other persons, like learners or tutors. Systems addressing these requirements must allow users to have a big flexibility in the way they use the system, how they collaborate, how they share their content, etc. Controlling who can access a document, specifying when a student may be contacted for interactive instant messaging or periodical reminders in order to increase motivation for collaboration are just some examples of typical statements that may be specified by e.g., learners and learning management system administrators. Policies represent an appropriate way for expressing this kind of statements because of their flexibility and dynamicity as well as their ease of use (they are typically declarative, that is, users specify "what" to do but not "how" something is to be done, therefore making its use more accessible to e.g., learners). Furthermore, lately there has been extensive research, that provides not only the ability of specifying these statements but also advanced mechanisms for reasoning on, exchanging and exploiting them.

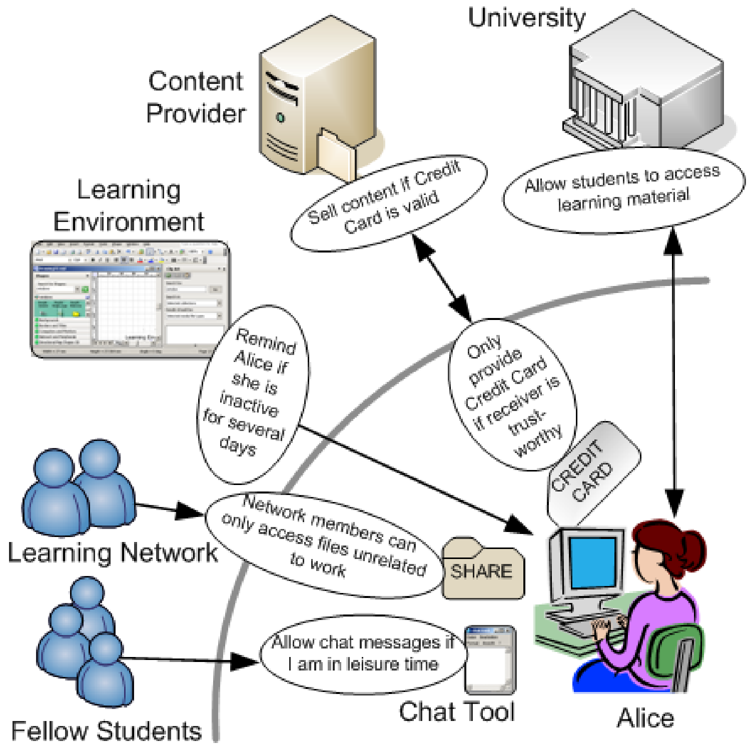
This paper focuses on the use of policies, a well-defined declarative and dynamic approach in order to specify and control the behavior of complex and rapidly evolving infrastructures for lifelong learning. First, Section 2 identifies example situations in which the specification of policies would increase the flexibility of the interactions and collaborations as well as enhance the learners experience. These examples show that dynamicity and ease of use, are a crucial requirement, both being two of the main characteristics of policies. The benefits of the integration of policies into learning management systems and personal learner agents in order to support such situations are described in Section 3, as well as the out-of-the-box benefits of their exploitation. In addition, Section 4 analyzes existing policy languages and frameworks in order to present an overview of available solutions to the reader. It provides a comparison of their main features as well as their advantages and disadvantages from the perspective of their integration into lifelong learning infrastructures. Later, Section 5 exemplifies the formalization of policies using a selected policy language and describes some of the added benefits of its use such as negotiations and advanced explanations. Finally, related work is presented in Section 6 and Section 7 concludes the paper.

2 Motivation Scenario

Alice holds a master degree in computer science and works successfully in a company. Recently, she was assigned the task of managing a new project starting in a couple of months and therefore she would need to learn and refresh her knowledge on project management. Since she has a full time job including many business trips, she uses an on-line learning client that allows her to improve her competences whenever she has some available time. With this learning client she is able to collaborate and to send questions or answers to other learners or tutors, and therefore she is able to chat with other students and even participate in a social network. However, since she uses her chat tool also for her job she restricts her chat facility in a way that during working time only business contacts and other employees of her company can start a conversation therefore allowing other students to contact her only in her leisure time. Of course, students trying to contact her during working time get a brief explanation of why a conversation is not possible at that very moment and which even indicates when Alice can be contacted.

Within the program Alice is following, she accesses different learning activities and objects through her learning client. Some of this material is free of charge but a couple of learning activities she is interested in are offered each one by a different content provider that sells it. Since the material is sold on a good price, she decides to purchase it. Each provider tells Alice that either she has to have an account or she has to provide a credit card for payment of the learning activity. For the first provider she does have an account and provides her username and password. Therefore she retrieves the requested material. However, she does not know the second provider and she must disclose her credit card. Alice protected her credit card in a way that it would only be disclosed to providers she may trust and the learning client provides a mechanism by which a content provider and Alice can trust each other even if they have not had any transaction in common before.

The learning client Alice is using allows her to share exercises and other relevant documents stored in her computer (e.g., using a peer-to-peer network [12, 10]) with other students following the same program or within the same learning network. She may even create some new material out of what she learned and her experience at work. She specifies which documents are to be shared and which conditions other student must fulfill in order to be able to retrieve it (e.g., be part of same program or be a tutor). In order to ensure the success of the students, the learning client includes a personalizable agent. Among other uses for this agent, Alice can create some guidelines in a way that the agent reminds her when she has to finish some learning activities or sends her an e-mail when she has been inactive for more than a week. Tutors can also attach these kinds of rules to the learning activities or programs in order to motivate their students or even to define some guidelines for on-line games that will be followed by their students [11]. Thanks to all these flexible facilities and all their personalization and configuration possibilities Alice is able to finish her program successfully.



University
Content Provider
Learning Environment
Sell content if Credit Card is valid
Allow students to access learning material
Remind Alice if she is inactive for several days
Only provide Credit Card if receiver is trust-worthy
CREDIT CARD
Network members can only access files unrelated to work
SHARE
Learning Network
Allow chat messages if I am in leisure time
Chat Tool
Fellow Students
Alice

Fig. 1. Example policies in an open and flexible lifelong learning infrastructure

3 Using Policies for Lifelong Learning

The term policy is generally defined as “statement specifying the behavior of a system” and it is intended to guide decisions and actions. Policies are encountered in many real world situations in our daily life like, for instance, shops may have a non return policy, with an exception for the week after Christmas, when a lot of unwanted items are returned. However, with the digital era, the specification of policies has emerged in many web-related contexts and software systems. E-mail clients filters are a specific kind of policy. One of the main application areas where policies have been lately used is security, privacy and the business domain (business rules). Policies yield many advantages compared to other conventional approaches. To mention some of them, policies are dynamic (and therefore can be easily change without recompiling the system that uses them), declarative (a step from the programmer closer to the user of a system since they state what to be done and not how it is to be done) and they typically have well-defined semantics and therefore they are easily exchangeable and understood by other parties, they usually allow reasoning over them (allowing to infer knowledge and not only explicit knowledge).

Considering the example from above (see also Figure 1), policies can support Alice in the situations described bringing in all the benefits of its use. Alice may formulate her personal preferences as a set of policies, for instance, her working hours or leisure time, as well as whom she considers as business contact or as a friend. Even a more powerful situation arises when Alice communicates with other parties. In our example, Alice uses the learning client to access content providers and receive access information or payment methods in a way her system can understand and process. Or she may easily define sophisticated methods to control who accesses the resources she has locally stored in her computer. In this situation, policies allow users and systems to characterize new users or systems by their properties and not simply by their identity (crucial in an open environment where completely strangers may interact with each other). For example, content providers may provide discount to students of a university (without having to know and update the whole list of students registered at the university) or Alice specifying that any user of a community she is in can access some of the resources in her computer or to whom she would disclose her credit card. Even negotiations can be (semi)automatically performed among entities driven by their policies [16].

Furthermore, policies can be used to control the behavior of software agents in order to send notifications, drive electronic games and simulations for educational purposes or many other approaches in order to increase her motivation while learning. All in all, policies have the potential to enhance the possibilities an open infrastructure offers while increasing its flexibility and ease of use.

4 Comparison of existing Policy Languages

This section first extracts requirements from our running example and identifies the features a policy framework should meet in order to address them. Then, it compares existing frameworks based on these features. The scenario presented above exploits policies in several ways like for example restrictions stated by Alice for incoming chat connections or conditions under which her locally stored resources (either documents or her credit card number) may be disclosed, content providers specifying whether a resource is free-of-charge or at cost (and the payment methods together with business rules like e.g., discounts) or general statements indicating how some entity (e.g., a software agent) should react to a specific event. An open lifelong learning infrastructure must provide sufficient functionalities in order to support all these situations. The following is a list of the most important features identified:

Positive vs. negative authorization: policies specifying conditions under which resources can be accessed may be of two types: positive or negative. Positive authorization policies specify that if the conditions are satisfied access is granted (e.g., "business contacts may start a conversation at any time") while negative authorization policies specify that if the conditions are satisfied access is denied (e.g., "if it is working time disallow students to contact Alice"). Although one may think that these two kinds of policies retain the natural way people express policies, it can be argued that the specification of negative authorizations complicates the enforcement of access control in a system [4] and adds the extra complexity of having to define metapolicies, that is, in case two policies conflict (one policy grants access and another denies it) a statement should specify which final decision is taken. Furthermore, this situation is scenario dependent. When dealing with security and access control, a typical approach is assuming that access to any resource is denied by default and only positive authorization policies are defined stating which resources are allowed [7, 2]. The reason is that if there exists an error in a policy, the cost of disclosing a sensitive resource is much higher than the cost of not disclosing a non-sensitive one.

Evaluation: it is important to allow policies to be private, that is, they are not disclosed unless some conditions are fulfilled. The reason is that policies may be sensitive. Imagine, Alice states that her friends can contact her via chat but not any of her business contacts. Most probably Alice does not want her business contacts to see this policy. Therefore, it is important to distinguish how policy evaluation is performed in different frameworks. Some of them assume that policies from different parties can be collected in a centralized place where a local algorithm is performed while other frameworks allow each entity to keep control of its policies without disclosing them to any other party and the algorithm for policy exchange and evaluation is distributed.

Negotiations: directly related to the previous requirement, the support for negotiations is another desirable feature in lifelong learning. Entities should be able to exchange policies with other entities at runtime. Since some policies may be private it may lead to negotiations like the on-line transaction between Alice and the content provider asking for her credit card[1].

Explanations: It should be possible to generate explanations out of the policies. On the one hand, they help a user to check whether the policies she created are correct and, on the other hand, they inform other users about why a decision was taken (or how the users can change the decision by performing a specific action). For example, if a student tries to contact Alice during her working time, that student would rather appreciate receiving a message like “I am not available from 8:00am to 5:00pm” instead of “I am not available”. Or if Alice discloses her credit card number to a content provider and it is not accepted a message like “This credit card is invalid because it is expired” would be more useful than simply “Invalid credit card”.

Strong/lightweight evidences: in many cases, entities have to provide information in order to satisfy other entities’ policies. In some cases, these properties need to be proved (e.g., Alice’s credit card or her student card number should be digitally signed for the content provider to accept it) and sometimes such strong evidence is not needed (e.g., Alice providing her e-mail address to the same content provider). It is important that a policy framework allows for both kinds of evidences.

Ontologies: as stated above, policies will be exchanged among entities within the lifelong learning infrastructure. Although the basic constructs may be defined in the policy language (e.g., rule structure and semantics), policies may be used in different applications and even define new concepts. Ontologies help to provide well-defined semantics for new concepts to be understood by different entities.

To date many policy languages have been developed. Among the most popular ones we can include KAoS [15], PeerTrust [7], Ponder [4], Protune [2], Rei [9], WSPL [1] and XACML [13]. The number and variety of languages is justified by the different requirements they were designed to accomplish. Ponder allows for local security policy specification and the description of security management activities like registration of users or logging and audit events to be used in the context of firewalls, operating systems or databases. WSPL’s name itself (namely Web-Services Policy Language) suggests its goal: supporting description and control of various aspects and features of a web service. Web services are addressed by KAoS too, as well as general-purpose grid computing authorization, although it was originally oriented to software agent applications (where dynamic runtime policy changes need to be supported). PeerTrust provided a simple but powerful language for performing negotiations on the Web, in peer-to-peer networks and on the grid, based on distributed query evaluations. Rei’s design was primarily concerned with supporting pervasive computing applications in which people and devices are mobile and use wireless networking technologies to discover and access services and devices). Protune’s broad notion of policy aims at

[1] Alice may require the content provider to provide a credential of the “Better Business Bureau” before she discloses her credit card, therefore leading to an iterative disclosure of policies and/or credentials

	KAoS	**Ponder**	**PeerTrust**	**Protune**	**Rei**	**WSPL**	**XACML**
Authorization types	Positive & negative	Positive & negative	Positive	Positive	Positive & negative	Positive & negative	Positive & negative
Negotiations			Y	Y			
Evaluation	Centralized	Centralized	Distributed	Distributed	Centralized	Centralized	Centralized
Explanations				Advanced			Simple text
Kind of Evidences			Strong & lightweight	Strong & lightweight	Strong		
Delegation		Y	Y	Y	Y		
Ontology support	Actions, actors...	Roles, groups...		Actions	Classes, properties...		
Open source implementation			Y	Y			Y

Table 1. Policy language comparison (Y = supported)

addressing any general application scenario, including e.g., trust management, security and privacy policies, business rules and quality of service specifications. Finally XACML was meant to be a standard general purpose access control policy language, ideally suitable to the needs of most authorization systems.

As shown in Table 4, Ponder, Rei, PeerTrust and Protune support delegation but only PeerTrust and Protune also allow for negotiations and both strong and lightweight evidences. However, Protune is the only policy language also supporting advanced explanation mechanisms and seems to be one the most complete language (as it is also demonstrated in [5]). On the other hand, Protune assumes by default that resources are private, therefore not allowing for the specification of negative authorizations, which is a feature supported by other frameworks like Rei or KAoS. However, Protune does not only allow for distributed evaluation of policies (therefore allowing policies to be kept private), but also open source implementations are available, making it easily accessible, usable and extendible.

5 Formalizing Policies in an Open Infrastructure

Previous sections describe the benefits obtained from the use of policies in an open system. In addition, they provide a description of some of the most important policy languages defined to date. In this section, we select the Protune policy language and, after a brief introduction to the language, we materialize some of the policies described in natural language in section 2 and present some of the added benefits that they provide, namely explanations and negotiations.

5.1 Protune language

In this section a brief overview of the Protune language is provided. Only the features which are required in order to support the scenarios we are interested in are described here. For an overall view of the language, as well as for a thorough description of its syntax and semantics see [2].

The Protune policy language is based on normal logic program rules

$$A \leftarrow L_1, \ldots, L_n.$$

where A is a standard logical atom (called the *head* of the rule) and $L_1, \ldots, L_n$ (the *body* of the rule) are literals, that is, L_i equals either B_i or $\neg B_i$, for some logical atom B_i.

In addition to usual Logic Programming-based languages, Protune provides support to actions, evidences and metapredicates.

Actions Protune allows to specify actions within a policy: typical examples of actions are sending evidences, accessing legacy systems (e.g., a database) or environmental properties (e.g., time). Actions are represented as usual predicates (called *provisional* predicates). Provisional predicates hold if they have been successfully executed

Evidences Protune allows to refer to evidences (i.e., credentials and declarations) from within a policy. Evidences can be regarded as a set of property-value pairs associated to an identifier. Each property-value pair is represented according to an object oriented-like dot notation $id.property : value$

Metapredicates Protune allows to define predicate properties. A *metapredicate* is a predicate associated with property-value pairs. They are represented through a notation close to the one used for evidences, namely $predicate \rightarrow property : value$. Rules containing metapredicates are called *metarules*. Metarules are typically exploited to assert some information about predicates occurring in a policy, e.g., the type of the predicate (property `type`) or some directives for the verbalization of the predicate which are meant to be used by the explanation facility (property `explanation`). Some properties apply only to provisional predicates: the value of the property `ontology` is the identifier of the action associated to the provisional predicate as reported in some ontology, whereas property `actor` (resp. `execution`) specifies which peer should carry out the action (resp. when the action should be performed)

In the rest of this section a policy fragment will be presented and explained. The fragment will be exploited in section 5.2 as well.

(1) $is\_colleague(Name) \leftarrow$
(1.1) $credential(C),$
(1.2) $C.type : employee,$
(1.3) $C.owner : Name,$
(1.4) $C.issuer : companyXYZ,$
(1.5) $C.public\_key : K,$
(1.6) $challenge(K).$

(2) $is\_colleague(\_) \rightarrow type : abbreviation.$

(3) $credential(\_) \rightarrow type : provisional.$
(4) $credential(\_) \rightarrow actor : peer.$

(5) $challenge(\_) \rightarrow type : provisional.$
(6) $challenge(\_) \rightarrow actor : self.$
(7) $challenge(K) \rightarrow execution : immediate \leftarrow$
$ground(K).$

This policy fragment contains a rule (1) and six metarules (from (2) to (7)). The rule states that the predicate *is_colleague* holds if each literal in the body of the rule holds.

- Metarules (2), (3) and (5) state that predicates *credential* and *challenge*, but not *is_colleague*, are provisional predicates
- Metarule (4) (resp. (6)) states that the action associated to *credential* (resp. *challenge*) must be performed by the other (resp. current) peer. In the following we assume that provisional predicate *credential* (resp. $challenge(K)$)

is associated to the action of sending a credential to the other peer (resp. checking whether the other peer is the holder of the public key K through a standard challenge procedure)
- Metarule (7) states that $challenge(K)$ can be executed if K is ground, i.e., instantiated

Assuming that we want to check whether Bob is a colleague, the policy fragment will be evaluated against the goal $is\_colleague('Bob')$ as follows

- line (1.1) checks whether a credential *cred*, has been sent by the other peer. If it is the case the evaluation proceeds, otherwise a failure is reported
- lines from (1.2) to (1.5) check whether the values of the properties of *cred* correspond to the ones listed in the body of the rule. If it is the case the evaluation proceeds, otherwise a failure is reported
- when evaluating line (1.6) the action associated to $challenge(key)$ is executed, where *key* is the public key of *cred*

5.2 Motivation Scenario (Revisited)

Our running scenario in section 2 contained many policies specified in natural language. In this section, we formalize them using the Protune policy language as a proof of concept of its power. It is important to note that users will not be requested to specify their policies in a rule-based logic language such as Protune. In contrary, end-users will be able to select and instantiate existing policies from a standard library[2] or, for advanced users, appropriate tools for the specification of new policies will be provided. In fact, most of the policy languages presented in section 4 provide management editors that help end-users and administrators to create and manage their policies.

In our running example Alice needs to specify that during work time her chat facility must only accept incoming messages from business contacts and other employees of her company.

Alice:

$$\begin{aligned}&allow(access(chat(Requester, init\_conversation))) \leftarrow\\&\quad working\_time(),\\&\quad is\_business\_contact(Requester).\end{aligned}$$

$$\begin{aligned}&allow(access(chat(Requester, init\_conversation))) \leftarrow\\&\quad working\_time(),\\&\quad is\_colleague(Requester).\end{aligned}$$

$$\begin{aligned}&allow(access(chat(Requester, init\_conversation))) \leftarrow\\&\quad leisure\_time().\end{aligned}$$

[2] E.g., similar to the mechanisms used in Microsoft Outlook for the instantiation of filtering rules.

$allow(access(chat(Requester, Action))) \rightarrow explanation:$
    $Requester$ & " $can\ contact\ Alice\ for\ action$ " & $Action.$

where *working_time*, *leisure_time*, *is_business_contact* and *is_colleague* may be defined as

```
Alice (contd.):
    $working\_time() \leftarrow$
        $time(T),$
        $T > 8:30, T < 17:00.$

    $leisure\_time() \leftarrow$
        $not\ working\_time().$
    $is\_business\_contact(Name) \leftarrow$
        $retrieve\_contact(Name),$
        $Name.category :' Business'.$

    $is\_colleague(Name) \leftarrow$
        $credential(C),$
        $C.type : employee,$
        $C.owner : Name,$
        $C.issuer : companyXYZ,$
        $C.public\_key : K,$
        $challenge(K).$

    $working\_time() \rightarrow explanation : $"It is working time".
    $leisure\_time() \rightarrow explanation : $"It is leisure time".
    $time(T) \rightarrow explanation : $"Time is " & $T.$
    ...
```

Specially important in this example is that using Protune, in case Bob, a friend that studies with Alice, tries to contact her during her working time, an explanation will be automatically generated from the specified policy [3]. Such explanation provides natural language statements such as

```
It can't be proved that
Bob can contact Alice for action init_conversation because
    there is no Requester such that
    Requester is a business contact              [details]
AND
    there is no Requester such that
    Requester is a colleague in companyXYZ       [details]
AND
    it is not true that
    It is leisure time                           [details]
```

In this explanation, statements that are made true and do not depend on the requester are hidden so the explanation is focused on the conditions which are not fulfilled (still full explanations providing such details can be generated too). In addition, clicking on *[details]* in a line provides a new explanation for the concept described in such a line. This way end-users may use hyperlinks in order to explore the proofs generated during the policy evaluation, from a more general description to a focused explanation of the concepts.

In addition, Alice also has the following policy protecting her credit card when trying to access on-line resources at different learning resource providers:

Alice (contd.):

$$\begin{array}{l} allow(access(CC)) \leftarrow \\ \quad CC.type : credit\_card, \\ \quad bbbMember(User). \\ \\ bbbMember(User) \leftarrow \\ \quad credential(C), \\ \quad C.issuer :' Better\ Business\ Bureau', \\ \quad C.name : User, \\ \quad C.public\_key : K, \\ \quad challenge(K). \end{array}$$

She finds a course she is interested in and requests access to it. The provider she is contacting has the following policy:

Learning Provider:

$$\begin{array}{l} allow(access(Course)) \leftarrow \\ \quad price(Course, Price), \\ \quad paid(User, Course, Price). \\ \\ paid(User, Resource, Price) \leftarrow \\ \quad credential(CC), \\ \quad CC.type : credit\_card, \\ \quad CC.owner : User, \\ \quad authenticated(User), \\ \quad charged(Resource, Price, CC). \\ \\ allow(release(credential(bbbCredential))). \end{array}$$

When Alice requests access to the course, these two policies raise a dynamic negotiation (as depicted in figure 2) allowing them to satisfy their respective policies in an iterative way and successfully perform such an on-line interaction although they were strangers and had not had any other transaction in common.

6 Related Work

To the best of our knowledge, using policy-based behavior control in technology-enhanced learning environments has not been extensively researched. An approach aiming at federated access control in web-service based repositories is

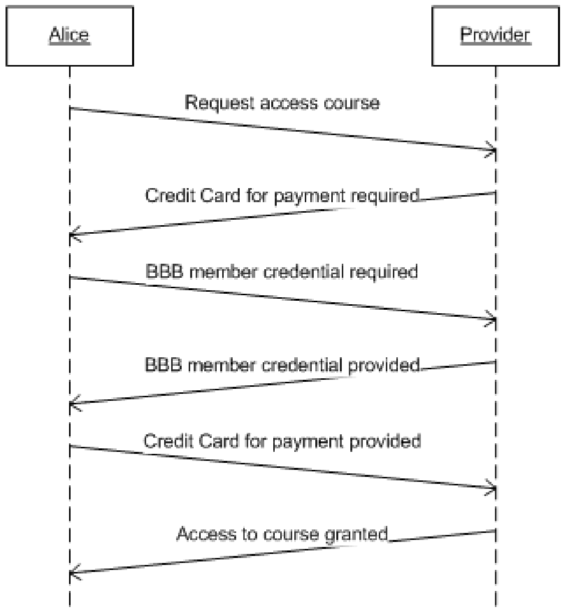

Alice
Provider
Request access course
Credit Card for payment required
BBB member credential required
BBB member credential provided
Credit Card for payment provided
Access to course granted

Fig. 2. Example negotiation sequence between Alice and the learning provider.

presented in [8]. In order to allow for an appropriate access control, the policy language XACML and the federated trust framework Shibboleth have been extended and integrated into an ECL middleware. In this framework, policies are based on a simple attribute directory service. In LionShare [10], there is a similar approach exploiting Shibboleth. Security is provided by so-called Access Control Lists expressed in XACML. These lists define which user can access which file depending on the users'properties, such as the membership of a certain faculty. However, none of these approaches allow for expressive access control supporting e.g., action executions, negotiations or explanations and therefore they do not meet the requirements identified in our scenario. Furthermore, using Shibboleth implies the existence of institutions users belong to - which is an assumption that does not apply in an open scenario for lifelong learning. [6] provides an abstract overview on privacy and security issues in advanced learning technologies, suggesting that policies may be used in an educational context. But neither scenarios nor specific details are provided. [17] deals with policies based on the Ponder policy language within the scope of collaborative e-learning systems. The use of policies in such a framework is basically restricted to role-based access control and therefore does not match the need of an open learning environment as described above.

7 Conclusions and Further Work

Open lifelong learning environments require flexible and interoperable approaches which are easy to use and to personalize by learners and tutors. This paper describes an advanced scenario for collaboration, exchange and utilization of learning resources. It also shows how policies can naturally address the requirements extracted from such a scenario providing benefits not only in flexibility and dynamicity but also additional features like reasoning and exchangeability. The paper gives an overview of existing policy frameworks and compares them according to the requirements previously identified. Finally, the paper shows how an existing policy language can be used in order to specify policies that may be used at runtime to e.g., control access to resources, perform negotiations or generate explanations.

As part of the TENCompetence project [14], which aims at supporting individuals, groups and organizations in lifelong competence development, we are currently working toward the exploitation of policies in order to increase and enhance the possibilities the TENCompetence infrastructure provides. According to the features provided by the different frameworks, we investigate the use Protune in order to support, among others, negotiations and advanced explanations, key issues for our open infrastructure.

Acknowledgements

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

References

1. Anne H. Anderson. An introduction to the web services policy language (wspl). In *POLICY '04: Proceedings of the Fifth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'04)*, page 189, Washington, DC, USA, 2004. IEEE Computer Society.
2. Piero A. Bonatti and Daniel Olmedilla. Driving and monitoring provisional trust negotiation with metapolicies. In *6th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2005)*, pages 14–23, Stockholm, Sweden, June 2005. IEEE Computer Society.
3. Piero A. Bonatti, Daniel Olmedilla, and Joachim Peer. Advanced policy explanations on the web. In *17th European Conference on Artificial Intelligence (ECAI 2006)*, pages 200–204, Riva del Garda, Italy, Aug-Sep 2006. IOS Press.
4. Nicodemos Damianou, Naranker Dulay, Emil Lupu, and Morris Sloman. The ponder policy specification language. In *POLICY '01: Proceedings of the International Workshop on Policies for Distributed Systems and Networks*, pages 18–38, London, UK, 2001. Springer-Verlag.
5. Claudiu Duma, Almut Herzog, and Nahid Shahmehri. Privacy in the semantic web: What policy languages have to offer. In *8th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2007)*, Bologna, Italy, June 2007. IEEE Computer Society.

6. El-Khatib, Korba, Xu, and Yee. Privacy and security in e-learning. *International Journal of Distance Education*, 2003.
7. Rita Gavriloaie, Wolfgang Nejdl, Daniel Olmedilla, Kent E. Seamons, and Marianne Winslett. No registration needed: How to use declarative policies and negotiation to access sensitive resources on the semantic web. In *1st European Semantic Web Symposium (ESWS 2004)*, volume 3053 of *Lecture Notes in Computer Science*, pages 342–356, Heraklion, Crete, Greece, May 2004. Springer.
8. Marek Hatala, Ty Mey (Timmy) Eap, and Ashok Shah. Unlocking repositories: Federated security solution for attribute and policy based access to repositories via web services. In *ARES '06: Proceedings of the First International Conference on Availability, Reliability and Security (ARES'06)*, pages 895–903, Washington, DC, USA, 2006. IEEE Computer Society.
9. Lalana Kagal, Timothy W. Finin, and Anupam Joshi. A policy language for a pervasive computing environment. In *4th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY)*, pages 63–, Lake Como, Italy, June 2003. IEEE Computer Society.
10. The lionshare project. `http://lionshare.its.psu.edu/`.
11. Thierry Nabeth, Claudia Roda, Albert A. Angehrn, and Pradeep Kumar Mittal. Using artificial agents to stimulate participation in virtual communities. In *IADIS International Conference CELDA (Cognition and Exploratory Learning in Digital Age)*, 2005.
12. Wolfgang Nejdl, Boris Wolf, Changtao Qu, Stefan Decker, Michael Sintek, Ambjörn Naeve, Mikael Nilsson, Matthias Palmer, and Tore Risch. Edutella: A P2P networking infrastructure based on RDF. In *11th International World Wide Web Conference (WWW'02)*, Hawaii, USA, jun 2002.
13. OASIS eXtensible Access Control Markup Language. `http://www.oasis-open.org/specs/index.php#xacmlv2.0`.
14. TENCompetence: building the european network for lifelong competence development. `http://www.tencompetence.org/`.
15. Andrzej Uszok, Jeffrey M. Bradshaw, Renia Jeffers, Niranjan Suri, Patrick J. Hayes, Maggie R. Breedy, Larry Bunch, Matt Johnson, Shriniwas Kulkarni, and James Lott. Kaos policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement. In *4th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY)*, pages 93–, Lake Como, Italy, June 2003. IEEE Computer Society.
16. William H. Winsborough, Kent E. Seamons, and Vicki E. Jones. Automated trust negotiation. DARPA Information Survivability Conference and Exposition, IEEE Press, Jan 2000.
17. Yang, Lin, and Lin. Policy-based privacy and security management for collaborative e-education systems. In *5th IASTED Multi-Conference Computers and Advanced Technology in Education*, Cancun, Mexico, 2002.