Bridging the Gap between User Attributes and Service Policies with Attribute Mapping

Davide Cerri and Francesco Corcoglioniti
CEFRIEL – Politecnico di Milano
Milano, Italy
{*cerri, corcoglioniti*}*@cefriel.it*

***Abstract*—People, companies, and public authorities can now have a strong on-line presence and a huge amount of interactions on the Internet, made possible by the impressive growth of the World Wide Web and of Web technologies. Many independent parties provide services and exchange information in a plural, dynamic, and open environment. This scenario, where interacting parties are often strangers, naturally brings to attribute-based access control solutions, as traditional identity-based systems are usually inadequate to large open environments. User attributes certified by external authorities, however, tend to be rather general-purpose and to reflect a user point of view, thus they often do not coincide with the concepts that are relevant for the service. In this paper we propose a framework to decouple the user point of view and the service point of view on user attributes: following our model, the service access control policy can focus on the concepts that are relevant for the service logic, whereas a separate attribute mapping policy establishes the bridge between the two domains.**

***Keywords*-user attributes; policies; attribute mapping; services; Semantic Web technologies;**

I. Introduction

The impressive growth of the World Wide Web and the evolution of Web technologies towards Web 2.0, the Web of Services and the Semantic Web, hugely multiplied the interactions between different actors, being them companies, people, or public authorities. This is an open, plural, and dynamic environment, where many independent parties provide services and exchange information, without any central coordination or control point.

In this scenario, interactions between users and services are often sporadic, and interacting parties are often strangers. Services typically need some information about users, e.g. in order to decide whether to permit or deny access to a certain feature or functionality, but it is not possible to assume that each service or application must have a-priori knowledge about all its potential users. It is therefore necessary to rely on third parties that vouch for user information, playing the role of *trusted authorities*. We can find many such authorities in the "real world": governments, professional associations, educational institutions and companies can certify information about users, in the form of *user attributes*. For example, a government can certify the name and citizenship of a person, a university can certify the possession of a degree, and an airline can certify frequent flyer status. Services can use this information to build a user profile (if needed) and to give the user appropriate privileges, according to the service policy. This situation thus naturally brings to *attribute-based* access control solutions and architectures, because traditional identity-based systems are inadequate to large open environments.

Relying on attributes and information coming from external, independent sources is however not so straightforward as it could seem at a first sight. Indeed, different authorities may use different attributes for the same or very similar piece of information, which can be considered equivalent for the service purposes. Moreover, the attributes issued by external authorities may not be exactly what the service would like to know. This is because in an open scenario in which attribute authorities are not meant for specific services and exist independently of any particular service, attributes are necessarily general-purpose[1]. Indeed, attributes certified by external third parties tend to be "on the user side" and to reflect a *user point of view*. On the other hand, the *service point of view* is often different, because the relevant concepts for the service are often not the same as the general-purpose user attributes.

Table I provides some simple examples of how the service point of view can be different from the user point of view. In a B2B scenario, an employee from the ACME company would probably have an assertion from her company which certifies that she is an "ACME employee", but for a business partner service the relevant information could just be that the user is a "partner company employee". In an e-health scenario, a doctor accessing information about a patient in another hospital may be seen by the service as a "non-local doctor" (e.g. because local and non-local doctors have different access privileges on patients' data), but this is probably not the attribute that the user's attribute authority certifies. Besides user attributes, some other external knowledge can be useful or needed to bridge the gap. An example can be given by an e-commerce Web site that has a separate wine section: the user attributes may say that the user is Italian

[1] An attribute authority established and used just for a particular service can actually be seen as a different form of service-specific user account.

Table I
USER VS. SERVICE POINT OF VIEW

User point of view	*Service point of view*
ACME employee	Partner company employee
Cardiologist at Niguarda hospital in Milan	Non-local doctor
Italian aged 25	Qualified to buy alcohol

and is 25 years old, whereas the relevant information for the service is that the user is qualified to buy alcohol. In this case, the legal age required to buy alcohol in different countries is an example of external knowledge used to translate the user attributes into the information needed by the service.

Writing the service access control policy with respect to the user point of view may not be straightforward. Moreover, having to deal with many concepts that may be far from being the most appropriate for the service needs, a policy written with respect to the user point of view can be unnecessarily long, complex, prone to errors, and difficult to maintain. Therefore, in this paper we propose a framework to decouple the user point of view and the service point of view on user attributes. Following our model, the service access control policy can focus on the concepts that are relevant for the service logic, whereas a separate attribute mapping policy establishes the bridge between the two domains.

The rest of this paper is organised as follows: after a brief survey of related technologies and work in section II, in section III we present our attribute mapping model, discussing also some implementation details in section IV. In section V we discuss some possible enhancements and future work, and we finally conclude in section VI.

II. RELATED WORK AND TECHNOLOGIES

In this section we briefly survey existing technologies and work that are relevant to our model. In particular, we consider access control and trust management models, Semantic Web technologies, and SAML.

It is widely recognised that traditional, identity-based, access control models are not adequate for large open and dynamic environments, where interacting parties are often strangers. Therefore, *Attribute Based Access Control* (ABAC) models have been proposed [1], [2], in which attributes other than identity are used in determining the trustworthiness of a party, and access control policies specify the attributes that a requester needs to possess in order to gain access to service or data. Directly using the attributes provided by the user in the access control policy, however, implies adopting what we called the user point of view in the service policy. The model we propose in this paper is coherent with an attribute-based vision, because it relies on attribute information rather than on identities. We however do not propose an access control model: our model acts as a bridge between the attribute-based user point of view and the access control logic of the service, independently of the access control model used by the service.

Even if we do not address the problem of trust (see section V), there is an overlap between our model and the area of *Trust Management* systems [3], [4]. RT [5] in particular is a well-known trust management language which, even if it has a much broader scope, could be used to address also the attribute mapping problem we identified. The basic concepts and syntax of RT are very general and expressive, but this has the drawback that, put in our context, RT does not provide a clear identification and separation of concepts that we consider relevant. We believe that a clear modelling of the attribute mapping problem and policy, as we propose in this paper, can be a valuable tool in tackling this problem and having clearer and more manageable policies. Moreover, the usage of Semantic Web technologies allows us to incorporate and reuse existing knowledge, without the need for the policy designer to explicitly encode this knowledge in an ad-hoc form in the access control policy.

Semantic Web [6] technologies aim at realising the vision of a Web of Data, made of interlinked resources described in a form directly understandable by machines (contrarily to regular Web pages). To this aim, technologies, approaches, and tools have been developed to effectively represent, integrate, query and reason over complex information, and the applicability of these instruments goes beyond the Web of Data scenario. RDF (Resource Description Framework) [7] is the Semantic Web data model, based on the organisation of information in ⟨*subject predicate value*⟩ triples. Knowledge representation languages, such as SKOS (Simple Knowledge Organization System) [8], and Ontology languages such as RDFS (RDF Schema) [9] and OWL (Ontology Web Language) [10] build on top of RDF and allow the formalisation of concepts, properties and instances that describe information in RDF. Ontology semantics is described in terms of description logic (DL), and reasoners are available to infer implicit knowledge. Rules can be supported by reasoners and specify what can be inferred starting from certain information. Several proprietary rule languages exists for the Semantic Web; a standard language does not exist yet, although RIF (Rule Interchange Format) [11] is undergoing standardisation by the W3C. A growing amount of machine-readable knowledge, in the form of ontologies, taxonomies, thesauri or simply RDF data is now published and available on the Web, ready to be exploited and integrated by (possibly unforeseen) applications.

Application of Semantic Web technologies to access control has been proposed in several works. In particular, Priebe et al. [12] address the problem of heterogeneity of attributes in ABAC systems and propose an extension of XACML (eXtensible Access Control Markup Language) that uses ontologies to model attributes and rules to map them from one ontology to another. In our model, we exploit Semantic Web technologies to map attributes similarly to [12], but

we propose a generic solution not bound to XACML. In particular, we use RDF as the underlying data model and a reasoner to implement the mapping rules. This approach has also the benefit to enable future extensions, as we discuss in section V.

The *Security Assertion Markup Language* (SAML) [13] is an OASIS standard that provides an XML-based framework for communicating user authentication, authorisation, and attribute information. It is often used to authenticate users in Web single sign-on scenarios; however it provides also attribute assertions. SAML attributes are name-value pairs, where the name can be e.g. a URI and the value can be any piece of XML data. SAML is an already widely deployed standard and it offers a significant flexibility; we therefore assume it as the format for user attribute assertions.

III. The attribute mapping model

A. Overview

In order to address the problems we discussed in section I, we propose to add an *attribute mapping* (AM) process before the regular access control logic of the service. The AM process does not substitute or interfere with the access control logic of the service: it works like a sort of adapter that takes care of the translation between the user point of view and the service point of view when receiving user attribute assertions, but it completely leaves access control decisions to the service business logic. The AM process is governed by its own policy (the *AM policy*), which is defined by the owner or administrator of the service.

The AM process translates the set of attributes provided by the user (user provided – UP – attributes) into a set of mapped attributes that refer to the service point of view (service mapped – SM – attributes) and that are then passed to the service access control logic. The attribute mapping allows users to use general-purpose attributes, with no need for an a-priori relationship with the service. On the other hand, the service access control policy can be written with respect to the service point of view, keeping the focus on the security needs of the service and the concepts that are relevant for it, and keeping the access control policy more compact and manageable.

Following the SAML model, we represent attributes as either names or name-value pairs. As prescribed by the Semantic Web vision, we use URIs as identifiers. Therefore, attribute names are URIs consisting of a namespace and a local name, where the namespace identifies the party defining the attribute, for example an attribute authority. Attribute values can be either literals, for specific data values (e.g. “John”, “25”, “black”), or URIs, for particular concepts or entities (e.g. http://dbpedia.org/resource/Milan or, shortly, dbpedia:Milan, to identify the city of Milan). The use of URIs permits to avoid name clashes and, more importantly, to create a bridge with other information sources describing the entities identified by those URIs. When only the name of an attribute is specified, the value is assumed to be unknown or not relevant. Attribute with multiple values can be represented by using several name-value pairs for the same attribute name; an “*and*” semantics is assumed in this case, i.e. the entity described by the attribute is assumed to be characterised by *all* the values specified for the attribute (e.g. a user with an attribute “driving licence” with values “car” and “truck” is assumed to have *both* a car and a truck driving licences).

An important property we want the AM process to satisfy is *monotonicity*. In the context of trust negotiation, Seamons et al. [14] define a policy language as monotonic if the disclosure of additional credentials (i.e., in our case, attributes) and policies can only result in the granting of additional privileges. Since the user decides which information to disclose, this property ensures that the user cannot get more privileges just hiding some information, i.e. not providing some attributes to the service. Our AM process only translates attributes from the user point of view to the service point of view, and does not take decisions about privileges to be granted. Nevertheless, if the “downstream” policy model is monotonic, the AM process must preserve this property, so that the process as a whole (from UP attributes to privileges) is monotonic according to the above definition. We therefore say that the attribute mapping process is monotonic if, given any set of UP attributes and the corresponding set of SM attributes (according to a given AM policy), any superset of this set of UP attributes corresponds to a (possibly coinciding) superset of the previously mentioned set of SM attributes. This means that if a set A_{UP} of UP attributes is translated into a set A_{SM} of SM attributes, every attribute in A_{SM} will remain true also if additional attributes are added to A_{UP}. Additional UP attributes can thus only result in additional SM attributes, so that, if the downstream model is monotonic, they can eventually only result in additional privileges.

As shown in figure 1, the mapping process we propose is conceptually organised in three consecutive steps[2]:

1) the *UP–UP step*, which uses a *user attribute taxonomy* to derive new attributes “on the user side” from given ones;
2) the *UP–SM step*, which maps from UP to SM attributes by means of *mapping rules*;
3) the *SM–SM step*, which uses a *mapped attribute taxonomy* to derive new SM attributes from existing ones.

The first step (*UP–UP*) aims at simplifying the input provided to subsequent steps. The user attribute taxonomy formalises the general case – particular case relation (subsumption) occurring between UP attributes, as seen by the service. For instance, the taxonomy can specify that “doctor” is a generalisation of (i.e. subsumes) “cardiologist”:

[2]This subdivision is only conceptual and does not prevent a different organisation at the implementation level.

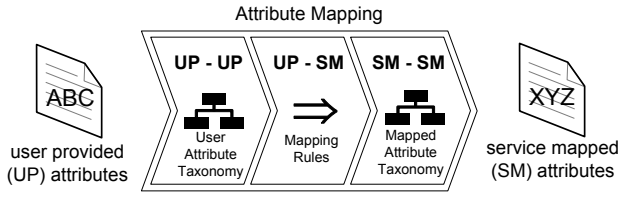

Figure 1. The attribute mapping process

this knowledge is then exploited in the UP–UP step to derive "doctor" each time "cardiologist" is encountered. This kind of derivation is especially useful to bring back to the same common concept similar attributes issued by different authorities, thus reducing the number of attributes to be considered in following steps. This mechanism allows also some decoupling between the rules of the UP–SM step (the core of the attribute mapping policy) and the input UP attributes, so that some changes in the latter can be addressed by simply updating the taxonomy.

The second step (*UP–SM*) is the core of the mapping process. Mapping rules have the form *IF body THEN head* (or, using a logical notation, $body \implies head$): the body states a set of conditions over UP attributes, considering both their existence and value, while the head states which SM attributes are derived if all the conditions in the body are satisfied. Thus, rules realise a bridge from the UP attribute domain to the SM attribute domain. Rules are designed to be evaluated in a *forward chaining* inference process, whose effect is to infer all the SM attributes that can be derived from available UP attributes according to the policy.

The third and last step (*SM–SM*) is similar to the UP–UP step and allows to make explicit and exploit generalisation relations that exist between SM attributes. These relations can be used to automatically derive general attributes from specific ones, like in the UP–UP step. As a consequence, these derived attributes do not need to be considered by mapping rules, which result simpler and more manageable.

The conceptual organisation of the mapping process into three steps does not increase the expressive power of the mapping, because taxonomy derivations could be represented as rules. We however believe that this organisation is clearer and helps the attribute mapping policy designer to express the relevant concepts in a clearer way. Although rules remain the main way to bridge the gap from the user to the service point of view, attribute taxonomies allow the designer to formalise and exploit the relationships between attributes. In particular, the user attribute taxonomy allows the designer to model the input of the mapping process; this taxonomy is especially useful to reconcile the mismatches between user attributes representing the same meaning in different ways. These mismatches can be caused by different authorities using different conceptualisations of a domain; for instance, different attributes can have the same meaning (synonyms) or some properties can be formalised both as attributes with values (e.g. *:gender : "Male"*) or by using multiple attributes without values representing classes of individuals (e.g. *:Male* vs *:Female*). Jointly, the two taxonomies allow to tackle a significant part of the overall mapping complexity, by reducing the number of attributes to be considered in mapping rules. As a result, rules are less, simpler and clearer, thus more manageable. Moreover, using attribute taxonomies the AM policy designer can import and thus reuse existing knowledge. This knowledge can be represented using Semantic Web languages (e.g. such as SKOS taxonomies or even OWL ontologies) and can be provided by external parties. For instance, it can represent shared and well known concepts of a particular application domain (e.g. in form of normative classifications published by standardisation bodies). In particular, taxonomies can be published by attribute authorities in order to document the kind of attributes they issue. In any case, it is important to note that the act of importing such knowledge must be explicit, due to the possible inferences deriving from it.

In the following we describe in details the two main instruments of the mapping process, i.e. mapping rules and attribute taxonomies, taking a car rental service as an example application. Figure 2 shows the mapping specification for this scenario.

B. Mapping rules

Mapping rules are at the core of the attribute mapping process. In order to formally define the semantics of rules we adopt a logical formalisation. Rules are represented by first-order definite Horn clauses, expressed in the form:

$$atom \wedge atom \ldots \wedge atom \implies atom$$

Atoms (i.e. atomic formulae) in the body of the rule represent conditions on UP attributes. Atoms in the head specify the SM attributes to be inferred if conditions are satisfied. In order to build atoms we define the following predicates:

- *Existence predicates*. For each attribute A we define a predicate A_{exist} with no arguments (i.e. a propositional variable); this evaluates to true *iff* the corresponding attribute is defined for the user (i.e., exists), regardless of the fact it has or not an associated value (or even multiple values).
- *Value predicates*. For each attribute A we define a unary predicate A_{value}; this evaluates to true for a given argument v *iff* the corresponding attribute exists and v belongs to the set of values associated to the attribute (this definition allows for attributes with multiple values).
- *Relational predicates*. We define predicates for commonly used binary relations, such as $=$, $<$, $>$ etc.

The use of predicates is constrained as follows:

- existence and value predicates in the rule body must refer to UP attributes, because they express conditions;

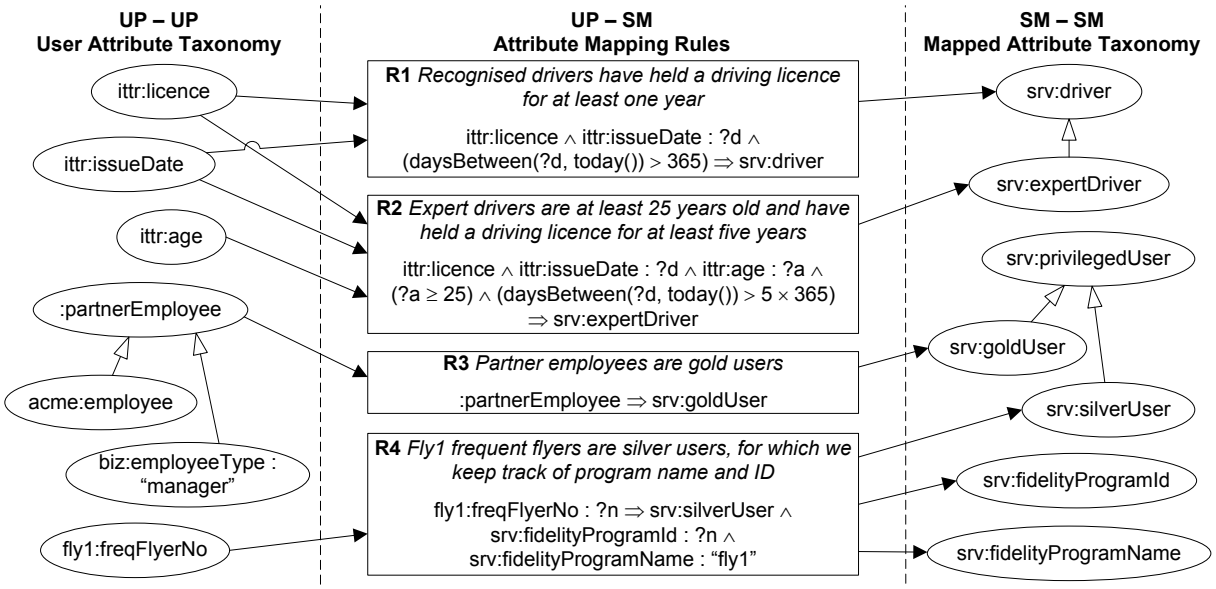

Figure 2. Example of attribute mapping declaration

- existence and value predicates in the head must refer to SM attributes, because they define derived attributes;
- relational predicates can only be used as conditions in the body part of the rule.

Predicates operate on *variables* and *terms*, as provided by first-order logic (FOL). Variables stand for values of UP attributes and are used to "join" conditions together and link the head to the body of rules; they are assumed to be implicitly universally quantified at the outer formula level and obey a safety condition, meaning that all variables in the head must also occur in the rule body. Terms, in turn, can be defined by composing constants and variables by means of general-purpose *functions*, such as arithmetic operators, and string and date manipulation operations. In order to simplify the notation, in the following and in the examples we represent the expressions A_{exist} and $A_{value}(v)$ with A and $A : v$ respectively; moreover, we adopt the infix notation for relational predicates and arithmetic operators, and we denote variables with the $?name$ pattern.

Figure 2 shows four examples of rules (R1 to R4) expressed both textually and according to the formalism detailed above. Consider for instance the mapping rule R1, stating that "*recognised drivers have held a driving licence for at least one year*". This rule can be formalised as follows:

$$\textit{ittr:licence} \wedge \textit{ittr:issueDate} : ?d \\ \wedge\, daysBetween(?d, today()) > 365 \implies \textit{srv:driver}$$

This rule can be read as: "IF *UP attribute ittr:licence is present* AND *UP attribute ittr:issueDate is present with a value ?d* AND *the distance in days between ?d and the current date is more than 365 days* THEN *SM attribute srv:driver is also present*". When used in a forward chaining rule engine, this rule will derive the SM attribute *srv:driver* whenever the conditions of its body are satisfied.

The use of definite Horn clauses gives an adequate expressive power while preserving an efficient runtime execution of rules, as we discuss in section IV. Negation as failure is not supported, together with other features (e.g. priorities) which are typical of logic programming but are not expressible in FOL. Therefore, the logic formalism we use falls under FOL, which is a monotonic logic. This means that learning a new piece of knowledge cannot reduce the set of what is already known, i.e. providing more UP attributes will never result in a smaller set of derived SM attributes. The inference performed using taxonomic relations can be also expressed through FOL axioms, therefore the monotonicity property we stated in section III-A is satisfied by the mapping process defined here.

When used in a forward chaining process, mapping rules do not allow for *recursion*, which happens when a rule, if activated, directly or indirectly triggers itself. More generally, a mapping rule cannot trigger the execution of another mapping rule, because of the strict separation between UP and SM attributes in the rule body and head. This is a precise design decision, which sacrifices a bit of expressiveness in order to enforce efficiency, comprehensibility and manageability of the rule base, avoiding complex chain effects that may be difficult to detect. This limitation does not however lead to a proliferation of mapping rules, thanks to the attribute taxonomies of the UP–UP and SM–SM steps, which enable a controlled form of "recursion" over attribute hierarchies.

C. Attribute taxonomies

Attribute taxonomies complement mapping rules and consider and exploit the general case – particular case relations (subsumption) occurring between UP and SM attributes, allowing the system to infer more generic attributes starting from available UP or SM attributes. Figure 2 shows a couple of taxonomic relationships for the car rental example:

- In the UP attribute domain, attribute *acme:employee* and attribute *biz:employeeType* with value *"manager"* are both traced back to the common, more general meaning of attribute *:partnerEmployee*. Since the policy designer is only interested in this meaning, he can write rules using *:partnerEmployee* and abstract from the particular UP attribute received (which can be *acme:employee* or *biz:employeeType*).
- In the SM attribute domain, users enjoying advantageous commercial treatment are identified by attributes *srv:privilegedUser*, *srv:goldUser* and *srv:silverUser*; the latter two are declared as special cases of the first one by the taxonomy, thus it is not necessary to extend rules R3 and R4 to derive also *srv:privilegedUser* each time that *srv:goldUser* or *srv:silverUser* are derived.

These examples highlight how the use of taxonomies on both the UP and SM sides allows to reduce the number of UP–SM mapping rules, because they decrease the number of UP and SM attributes that need to be considered by these rules.

In the example of figure 2 we provide a graphical, intuitive representation of taxonomies. More formally, a taxonomy can be expressed as a set of generalisation ($\sqsubseteq$)[3] relations between attributes, where the general semantics of $A \sqsubseteq B$ is that B meaning is more general than A (i.e. B subsumes A) and therefore we can derive B each time we encounter A. We deem useful to consider also values in this process, in order to improve expressiveness and being able to express relationships such as the one between attributes *biz:employeeType* and *:partnerEmployee* in figure 2. Using values requires in the general case to consider relations between name-value pairs, but since values are optional we can distinguish among four possible generalisation relations, whose semantics is given as follows:

- $A \sqsubseteq B$ specifies that attribute B is more general than attribute A; i.e. $A \implies B$ for attributes without values (a "*sub-class*" relation, e.g. *:cardiologist* $\implies$ *:doctor*) and $A : ?v \implies B : ?v$ for attributes with values (a "*sub-property*" relation, e.g. *:managerOf : ?company* $\implies$ *:employeeOf : ?company*). Note that in the latter case the range of allowed values for A must be a subset of the corresponding range for B (this is consistent with the semantics of a sub-property).

[3]We borrow the notation used in Description Logic to specify class and property inclusion.

Table II
EXTRACTING TAXONOMIES FROM RDFS AND SKOS CONSTRUCTS

RDFS construct	*Taxonomy relation*
P rdfs:domain C	$P \sqsubseteq C$
P rdfs:subPropertyOf Q	$P \sqsubseteq Q$
C rdfs:subClassOf D	$C \sqsubseteq D$
A skos:transitiveBroader B	$B \sqsubseteq A$
A skos:transitiveNarrower B	$A \sqsubseteq B$
A skos:exactMatch B	$A \sqsubseteq B$, $B \sqsubseteq A$

Note: P and Q are properties, C and D classes, A and B can be both properties or classes.

- $A : x \sqsubseteq B$ specifies that attribute A with value x has the same or a narrower semantics than attribute B, i.e. $A : x \implies B$ (e.g. *icd-standard:diseaseCode : "H54.5"* $\implies$ *:visualImpaired*).
- $A \sqsubseteq B : y$ specifies that attribute A has the same or a narrower semantics than attribute B with value y, i.e. $A \implies B : y$ (e.g. *universityX:EnrolledStudent* $\implies$ *:profession : "student"*).
- $A : x \sqsubseteq B : y$ specifies that attribute A with value x corresponds or has a narrower semantics to attribute B with value y, i.e. $A : x \implies B : y$ (e.g. *ittr:licence : "Classe B"* $\implies$ *us:drivingLicence : "Class C"*).

The use of taxonomies provides the capability to import existing knowledge, easing the task of the designer while still achieving good runtime efficiency, due to their simplicity and straightforward mapping to rules. Hereafter, we briefly describe how to translate RDFS ontologies and SKOS taxonomies into our taxonomies (a translation is required due to the peculiarities of the SAML-based attribute model, such as optional values). These languages can be used to specify the classes and properties characterising a user in a certain application domain. Properties map straightforwardly to attributes with values, while classes (of users) can be assimilated to attributes without a value (because only the information that the user belongs to a class must be conveyed). Under these correspondences, table II lists some conversion rules to apply in order to map RDFS and SKOS constructs to our taxonomic relations (note that we map only the constructs that permit to derive new attributes, i.e. to derive new facts at the A-Box level).

IV. IMPLEMENTATION

Attribute mapping can be implemented using Semantic Web technologies. This allows to exploit tools and results developed so far in this field and prospect new opportunities to evolve the AM system. Attributes are converted internally in RDF, by deriving a triple $\langle S\ A\ v \rangle$ for each attribute $A : v$, where the subject corresponds always to the user and a blank node or a special *"unknown"* value can be used when there is only the attribute name. Mapping rules and taxonomies are translated into inference rules, which are fed to a reasoner that infers new triples for derived SM attributes, by operating in forward chaining mode. The whole process is shown in

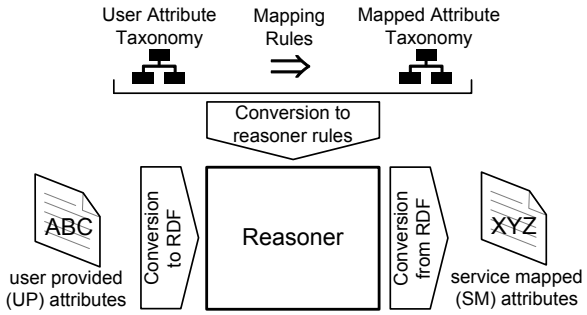

Figure 3. Attribute mapping implementation

figure 3. Following we report an example of rule encoded in RIF (rule R4 of our example) to get an idea of a concrete rule representation:

```
Prefix(...)

Forall ?s ?l ?d ?a (
  ?s[srv:expertDriver -> 'true'] :- And (
    ?s[ittr:licence ?l]
    ?s[ittr:issueDate ?d]
    ?s[ittr:age ?a]
    External(
      pred:numeric-greater-than-or-equal(
        func:subtract-dates(am:today(), ?d)
        func:numeric-multiply(5, 365)
      )
    )
    External( pred:numeric-greater-than-or-equal(?a 25) )
  )
)
```

Attribute mapping performance depends on several factors, including rule complexity and the number of UP attributes. Our requirements are however limited. First, we do not require persistence, which is often costly. Then, the chosen logic formalism (Horn logic) trades expressiveness for efficiency: since costly description logic constructs are not supported (only the DLP branch of description logic is supported), complex reasoning capabilities are not required and efficient rule evaluation algorithms can be employed. In particular, the RETE algorithm [15] at the basis of several reasoners seems appropriate to the AM process. RETE compiles rules into a network, whose nodes evaluate simple conditions, match variables and perform joins on them. The compile operation is costly, but it can be performed off-line because our rules do not change at runtime (and when they change a batch re-compilation can be triggered). RETE may also suffer of memory consumption problems when used with large knowledge bases, but this is clearly not our case because our rules operate just on the set of UP attributes. Therefore, a RETE-based reasoner can be effectively applied to our scenario.

V. Extensions and future work

Semantic Web technologies can be further exploited to extend the mapping approach we described in section III along multiple directions. First of all, their capability to represent, integrate and reason over complex data can be exploited to deal with complex attribute values, like the complex structures allowed by SAML. Secondly, ontology languages can be used to better specify and constrain the input and output of the mapping process. Input constraints specify the form of valid UP attributes (e.g. attribute type, minimum and maximum cardinality, disjointness constraints) and can be checked at runtime in order to avoid ambiguous situations that may arise when rules are fed with data they were not designed to handle. Output constraints specify the desired characteristics of SM attributes, and we expect them to be checked at design time (rather than at runtime) by a policy analysis tool. Lastly, integration of external knowledge can be supported to a greater extent than described in section III for taxonomies. In particular, mapping rules and supporting data (e.g. code conversion tables) are already available in many application fields, and formalised using Semantic Web languages. The reuse of this knowledge in the AM policy will obviously ease the task of the designer, but can degrade the performance of the system if a lot of rules or data needs to be considered at runtime when mapping user attributes. We argue that often just a small portion of this knowledge is relevant for a given AM policy (i.e., just a few mapping rules), but we cannot require the designer to specify it manually, thus an automated approach to select the relevant knowledge is needed.

Policy analysis is another possible extension, enabled by the logical formulation of the mapping policy. With this approach rules can be treated as axioms, while constraints and desired policy properties can be formalised as theorems to prove starting from this knowledge. FOL model checkers can be employed to analyse small policies, and we expect that less expressive formalisms, such as description logic, can make limited forms of analysis feasible even on large policies, e.g. to identify not assignable attributes or implicit subsumption relations among SM attributes (similar techniques have been already proposed in [16] for XACML policies).

Our model is focused on attribute mapping, while we currently have not considered the problem of trust, i.e. basically deciding whether to accept or not an attribute assertion depending on which authority issued it. As a first approximation, we can assume that trusted authorities are directly known by the service, hence trust decisions can be fairly simple. This assumption needs however to be weakened in large-scale open systems, where the service can rely on a set of known trust anchors but it may be unreasonable to require that all trusted authorities must be known a-priori. The already mentioned field of trust management systems provides solutions to this problem, which could be combined with our model in order to "filter" user attributes before mapping them to the service domain, exploiting relationships between different authorities (e.g. some sort of delegation). Such relationships can however

introduce the need for another kind of attribute mapping, i.e. between different authorities, using information (mapping rules and constraints) that should probably come from the authorities themselves. We therefore plan to extend our approach in order to cover also the trust problem in a coherent framework, with the related mappings. In this respect we deem interesting to follow a negotiation approach [14] (in order to address credential sensitivity and user privacy), and investigate the possibility of an automated "backward mapping" from attribute requirements expressed with respect to the service point of view to corresponding requirements expressed from the user point of view.

VI. Conclusion

Attribute-based security models appear more adequate than traditional identity-based models to address the challenges presented by a global, open, and dynamic environment like the Web, where interacting parties often do not previously know each other. The attributes that describe users must however be certified by third parties, and in a large-scale open environment attributes certified by external entities are usually general-purpose, thus they often do not represent very well the concepts that are actually relevant for the service access control policy.

In this paper we presented an attribute mapping model that provides a solution to this problem, decoupling the user point of view reflected in user attributes and the service point of view reflected in the service access control policy. Following our approach, the service access control policy can focus on the concepts that are relevant for the service logic, whereas a separate attribute mapping policy establishes the bridge between the two domains, providing a mapping from the attributes provided by the user to service-related concepts. Moreover, the attribute mapping process can be enriched with knowledge coming from other sources, which better describes the user domain and helps the translation into the service domain.

Acknowledgements

This work has been supported by the European Commission under the TripCom project (IST-4-027324-STP). The authors wish to thank their colleagues Emanuele Della Valle and Alessio Carenini for their suggestions and support.

References

[1] P. Bonatti and P. Samarati, "Regulating service access and information release on the Web," in *CCS '00: 7th ACM Conference on Computer and Communications Security*, Athens, Greece, Nov. 2000, pp. 134–143.

[2] E. Yuan and J. Tong, "Attributed Based Access Control (ABAC) for Web Services," in *ICWS '05: 3rd IEEE International Conference on Web Services*, Orlando, FL, USA, Jul. 2005, pp. 561–569.

[3] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized Trust Management," in *S&P '96: 1996 IEEE Symposium on Security and Privacy*, Oakland, CA, USA, May 1996, pp. 164–173.

[4] S. Weeks, "Understanding Trust Management Systems," in *S&P '01: 2001 IEEE Symposium on Security and Privacy*, Oakland, CA, USA, May 2001, pp. 94–105.

[5] N. Li, W. H. Winsborough, and J. C. Mitchell, "Distributed Credential Chain Discovery in Trust Management," *Journal of Computer Security*, vol. 11, no. 1, pp. 35–86, Feb. 2003.

[6] N. Shadbolt, T. Berners-Lee, and W. Hall, "The Semantic Web Revisited," *IEEE Intelligent Systems*, vol. 21, no. 3, pp. 96–101, May 2006.

[7] F. Manola and E. Miller, "RDF Primer," W3C Recommendation, Feb. 2004. [Online]. Available: http://www.w3.org/TR/REC-rdf-syntax

[8] A. Miles and S. Bechhofer, "SKOS Simple Knowledge Organization System Reference," W3C Candidate Recommendation, Mar. 2009. [Online]. Available: http://www.w3.org/TR/2009/CR-skos-reference-20090317/

[9] D. Brickley and R. V. Guha, "RDF Vocabulary Description Language 1.0: RDF Schema," W3C Recommendation, Feb. 2004. [Online]. Available: http://www.w3.org/TR/rdf-schema/

[10] D. L. McGuinness and F. van Harmelen, "OWL Web Ontology Language Overview," W3C Recommendation, Feb. 2004. [Online]. Available: http://www.w3.org/TR/owl-features/

[11] H. Boley, M. Kifer, P.-L. Patranjan, and A. Polleres, "Rule interchange on the web," in *Reasoning Web*, ser. LNCS, vol. 4636. Springer, Sep. 2007, pp. 269–309.

[12] T. Priebe, W. Dobmeier, C. Schläger, and N. Kamprath, "Supporting Attribute-based Access Control in Authorization and Authentication Infrastructures with Ontologies," *Journal of Software (JSW)*, vol. 2, no. 1, pp. 27–38, Feb. 2007.

[13] S. Cantor, J. Kemp, R. Philpott, and E. Maler, "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0," OASIS Standard, Mar. 2005. [Online]. Available: http://docs.oasis-open.org/security/saml/v2.0/

[14] K. E. Seamons, M. Winslett, T. Yu, B. Smith, E. Child, J. Jacobson, H. Mills, and L. Yu, "Requirements for Policy Languages for Trust Negotiation," in *POLICY '02: 3rd International Workshop on Policies for Distributed Systems and Networks*, Monterey, CA, USA, Jun. 2002, pp. 68–79.

[15] C. L. Forgy, "Rete: A Fast Algorithm for the Many Pattern / Many Object Pattern Match Problem," *Artificial Intelligence*, vol. 19, no. 1, pp. 17–37, Sep. 1982.

[16] V. Kolovski, J. Hendler, and B. Parsia, "Analyzing web access control policies," in *WWW '07: 16th International Conference on World Wide Web*, Banff, AB, Canada, May 2007, pp. 677–686.