

# A Semantic-based User Privacy Protection Framework for Web services <sup>\*</sup>

Arif Tumer<sup>2</sup>, Asuman Dogac<sup>1</sup>, and I. Hakki Toroslu<sup>1</sup>

<sup>1</sup> Software Research and Development Center &  
Dept. of Computer Eng.  
Middle East Technical University (METU)  
06531 Ankara Türkiye  
asuman@srdc.metu.edu.tr, toroslu@ceng.metu.edu.tr  
<sup>2</sup> Intro Solutions  
Ankara Türkiye  
arif.tumer@introsolutions.com

**Abstract.** Web service technology is an Internet-based distributed computing paradigm to address interoperability in heterogeneous distributed systems. In this paper, we present a privacy framework for Web services which allow user agents to automatically negotiate with Web services on the amount of personal information to be disclosed on behalf of the user. In developing this framework, the following key privacy considerations are taken into account: revealing only the minimal pertinent information about the user, not to overwhelm the users while declaring their privacy preferences and requiring only limited user interaction.

In the framework proposed, the Web services declare their input parameters as *Mandatory* or *Optional* and allow users to declare how much of their personal information can be made available to the services. The users specify their privacy preferences in different permission levels on the basis of a *domain specific service ontology* based on DAML-S. The major components of the system are a globally accessible context server which stores user preferences and a service registry where the services advertised and the service semantics are available.

## 1 Introduction

Currently Web services have become a main thrust of the IT industry: Many packaged application vendors, such as SAP, Siebel, and PeopleSoft are moving towards providing Web service APIs; others are building tools that export legacy mainframe applications in Web services form, while still others are making important functionality for business partners available through Web services [4] (e.g., the Amazon.com Web Service [1] and Google Web service API [14]).

---

<sup>\*</sup> This work is supported by the European Commission through IST-1-002104-STP Satine project and in part by the Scientific and Technical Research Council of Turkey, Project No: EEEAG 102E035

Considerable progress has been made in the area of Web service description and invocation: There are two almost universally accepted standards for these purposes: SOAP (Simple Object Access Protocol) [22] for invoking services and WSDL (Web Services Description Language) [30] for describing the technical specifications of the services. There are also two well-known service registries, UDDI [23] by Microsoft and IBM and ebXML [12] by UN/CEFACT.

Well accepted standards like WSDL and SOAP make it possible only to “dynamically access” to Web services in an application. That is, when the service to be used is known, its WSDL description can be accessed by a program which uses the information in the WSDL description like the interface, binding and operations to dynamically access the service.

In order to exploit the Web services to their full potential, their semantics must also be available. There is an important initiative in this respect, namely, DAML-S [7] which defines an upper ontology for describing the semantics of Web services. There are also efforts to complement this upper ontology with domain specific service ontologies such as the bioinformatics ontology given in [28].

Describing the semantics of Web services improves the Web service technology in several respects such as being able to define the properties of services like their real life counterparts and facilitating automated service discovery and composition.

Another area that can benefit from the semantics of Web services is protecting user’s privacy when accessing the Web services. There are some important considerations in developing privacy mechanisms:

- Only the minimal pertinent information should be provided to the Web service to prevent disclosing unnecessary personal information. As an example, a user may have to provide her credit card number when invoking a “purchasing” service but may prefer not to so for example for a “reservation” service.
- Another critical issue is not to overwhelm the users while declaring their privacy preferences. Indeed declaring privacy preferences on the basis of service instances may be quite cumbersome and sometimes even not possible. A user may not in advance know which service she will need.
- The process should be automatic requiring minimal user interaction. Current privacy management mechanisms like P3P [6] are oriented towards Web browsing and thus require user interaction.

In this paper we address protecting the users’ privacy when using Web services by addressing the issues mentioned above. We allow Web services to declare their input parameters related with personal user information as *Mandatory* or *Optional*. We show how DAML-S Service Profile input parameter specification [7] can easily accommodate the changes to differentiate between input parameters that are essential for the service to execute from those which are optional. Optional parameters are those a service provider is requesting for its own use.

The services also declare alternate data element requests in case that a user does not want to provide some mandatory input parameters. For example, if a

contact address is necessary for the service and the user is not giving her mobile phone number, her email address may be requested instead.

We allow users to declare how much of their personal information can be made available to the services. The users declare their privacy preferences as *Free*, *Limited*, or *NotGiven* on the basis of a domain specific service ontology. As an example, assume a service ontology in the “travel” domain (Figure 5). A “Hotel Reservation” service (a node in the ontology) may require the user’s name, email address, and date of reservation as *Mandatory* and a credit card number as *Optional*. A user on the other hand, may declare that for any hotel reservation service, she will provide all the requested information mentioned as *Free* but will not provide her credit card number (*NotGiven*).

The approach presented has the following advantages:

- The user preferences are defined not for individual service instances but for a node in the service ontology and thus applies to all instances unless the user overrides this explicitly. Furthermore, a user may declare the same policy for several different service classes. The effort required by the user is further minimized since the privacy preferences at the upper level classes of the ontology are inherited by lower level service classes. Note that a user can override a privacy preference at any level she likes.
- The presented framework allows Web services to declare alternate data requests if a mandatory input is not given by the user. This provides flexibility and creates room for reaching an agreement through negotiation.
- Finally, we believe that declaring the user preferences based on a standard service ontology like DAML-S helps with the interoperability problem.

The work accomplished in this paper is realized within the scope of the IST-1-002104-STP Satine Project [20],[11]. Satine aims to develop a semantic Web service based interoperability framework for tourism industry. In order to semantically annotate Web services, travel domain ontologies like Harmonise [15] are used. The project also proposes Web service ontologies based on the recent travel industry message specifications produced by Open Travel Alliance (OTA) [21].

The paper is organized as follows: In Section 2, a privacy framework for Web services based on domain specific service ontologies is presented. Section 3 describes the negotiation process between the user agent and the Web service. In Section 4, an example scenario is given to illustrate the concepts. Section 5 describes the related work. Finally, Section 6 concludes the paper.

## 2 A Privacy Framework for Web Services

We propose the following basic elements for the privacy model of the Web services:

- *Policy Statement*: Web services describe their business practices regarding the use of personal user information in *Policy Statements* - or *Service Policies*.

- *Input Requests*: The data set requested from the user is available as the input parameters of a Web service.
- *Privacy Preferences*: The user is responsible for declaring her *Privacy Preferences* regarding the policy statements and input parameters of Web services. The declaration of privacy preferences is based on a domain specific service ontology.
- *Service Request Analyzer*: This component is responsible for parsing and analyzing data request files and the policy statements that would be supplied by the service provider. The relevant rules in these files are converted into internal representations appropriate for the applications that will manage the negotiation process.
- *Rule Extractor*: This component determines the user’s privacy rules regarding a Web service, to be utilized during negotiation between the user agent and the service. Rule extraction is explained in Section 3.1.
- *Negotiation Component*: Based on the rules declaring the service’s request and rules describing the privacy preferences of the user, the negotiation component tries to find a ground for agreement between the user agent and the Web service. Comparing the necessity rules and permission levels and employing the alternatives, this component generates an *agreed-upon element set* that describes the elements that will be sufficient for the enactment of the service and allowed by the user. Negotiation mechanism is presented in Section 3.2.

## 2.1 Declaration of Privacy Policy

Policy statements provide a way to describe the data use practices. Web services should also declare their policies regarding their input parameters such as their purpose in requesting the data, with whom they may share the data with and whether and how they will retain data. For this purpose, we adapted the P3P policy mechanism to Web services as follows:

- *Purpose* section describes the basis for collecting user’s data,
- *Recipient* section declares the entities with whom the data will be shared,
- *Retention* section defines the activity scope during which the data will be retained.

A possible mechanism for Web services to declare such policies is to exploit DAML-S Service Profile input parameter. DAML-S service profile describes “what the service does”; that is, it gives the type of information needed by a service-seeking agent to determine whether the service meets its needs, typically such things as input and output types, preconditions and postconditions, and binding patterns [7].

We propose to specify the Web service privacy policies in DAML-S as shown in Figure 1.

Disputes and remedies are not incorporated into this work, as they are more high level statements, accompanied by textual descriptions, hence not much

```

<rdf:Property rdf:ID="input">
  <rdfs:subPropertyOf rdf:resource="#process;#parameter"/>
  <rdfs:domain rdf:resource="#service;#ServiceProfile"/>
  <rdfs:range rdf:resource="InputSpecs"/>
</rdf:Property>

<rdf:Property rdf:ID="purpose">
  <rdfs:subPropertyOf rdf:resource="#input"/>
  <rdfs:domain rdf:resource="InputSpecs"/>
  <rdfs:range rdf:resource="#daml;#Thing"/>
</rdf:Property>

<rdf:Property rdf:ID="recipient">
  <rdfs:subPropertyOf rdf:resource="#input"/>
  <rdfs:domain rdf:resource="InputSpecs"/>
  <rdfs:range rdf:resource="#daml;#Thing"/>
</rdf:Property>

<rdf:Property rdf:ID="retention">
  <rdfs:subPropertyOf rdf:resource="#input"/>
  <rdfs:domain rdf:resource="InputSpecs"/>
  <rdfs:range rdf:resource="#daml;#Thing"/>
</rdf:Property>

```

Fig. 1. Web Service Privacy Policies in DAML-S

machine-processable. A further point to note is the following: our privacy framework allows for *Purpose* and *Recipient* policies to be defined within multi level taxonomies [16] since subtypes are necessary for expressing more fine-grained policies.

There are two basic elements in the privacy model: the data requested by Web services and the privacy preferences of the users. These issues are discussed in the following subsections.

## 2.2 Specifying Data Requests of Web services

We define the data requested by a Web service to be composed of three parts: the first one is the set of elements requested by the Web service (that is, the input parameters of the service). The second one is the declaration of how essential the data is for the service to execute. Finally, a Web service may also provide rules stating alternate data elements if a mandatory piece of information is not provided by the user. For example a rule may state that if a user is not willing to disclose her email address, she should provide her postal address. Alternatives may help to reach an agreement during negotiation.

We use DAML-S service profile input parameter definition to specify whether the input parameter is essential for the service to execute (i.e., mandatory) or it is requested for some other purpose (i.e., optional) as shown in Figure 2.

We define alternative data requests through *Conditional Request* and express them in RDF. A conditional request is an “if-then” rule describing what alternate data elements may be of use if some mandatory data elements are not given by the user for a specific service class. Example conditional statements are presented

```

<rdf:Property rdf:ID="mandatory">
<rdfs:subPropertyOf rdf:resource="&process;
                    #inputParameter"/>
</rdf:Property>
<rdf:Property rdf:ID="optional">
<rdfs:subPropertyOf rdf:resource="&process;
                    #inputParameter"/>
</rdf:Property>

```

Fig. 2. Defining the types of Data Element Requests through DAML-S

in Figure 3. These rules state that if the user is not providing an “EmailAddress” and a “CreditCardNo” which are essential for the service to execute then, the user should provide a postal address and a bank account number, respectively.

```

<pri:IfRule>
  <pri:If rdf:parseType="Resource">
    <pri:NotGiven rdf:resource="...#emailAddress"/>
  </pri:If>
  <pri:Then rdf:parseType="Resource">
    <pri:Mandatory rdf:resource="...#postalAddress"/>
  </pri:Then>
</pri:IfRule>
<pri:IfRule>
  <pri:If rdf:parseType="Resource">
    <pri:NotGiven rdf:resource="...#creditCardNo"/>
  </pri:If>
  <pri:Then rdf:parseType="Resource">
    <pri:Mandatory rdf:resource="...#bankAccountNo"/>
  </pri:Then>
</pri:IfRule>

```

Fig. 3. Example Conditional Statements

### 2.3 Describing User Privacy Preferences

The users define their privacy preferences for Web services through a rule-based mechanism with a reference to a domain specific service ontology.

There are three permission level rules that can be imposed on a data element for a given service class:

- *Free* The data element is given freely by the user.
- *Limited* The data element is provided by the user only if it is mandatory for the service enactment.
- *NotGiven* The given data element is not provided by the user.

As an example consider the rule segment given in the Figure 4 and an example travel ontology given in Figure 5. The first rule is associated with the top level

```

<pri:Rule>
  <pri:Role rdf:resource="../../../TravelService"/>
  <pri:Data rdf:parseType="Resource">
    <pri:Limited rdf:resource="../../../age"/>
    <pri:Free rdf:resource="../../../home.Phone"/>
  </pri:Data>
</pri:Rule>
<pri:Rule>
  <pri:Role rdf:resource="../../../TransportationService"/>
  <pri:Data rdf:parseType="Resource">
    <pri:NotGiven rdf:resource="../../../creditCardNo"/>
    <pri:NotGiven rdf:resource="../../../mobile.Phone"/>
    <pri:Limited rdf:resource="../../../emailAddress"/>
    <pri:Free rdf:resource="../../../name"/>
  </pri:Data>
</pri:Rule>
<pri:Rule>
  <pri:Role rdf:resource="../../../BuyTicket"/>
  <pri:Data rdf:parseType="Resource">
    <pri:Free rdf:resource="../../../creditCardNo"/>
  </pri:Data>
</pri:Rule>

```

Fig. 4. User Context Privacy Rules

service class defined in this ontology (Figure 5). For the top level class, the user releases her home phone number freely but her age information is limited (that is it will be provided only if the service declares it to be mandatory). Through the second rule associated with the “TransportationService”, the user does not give her credit card number and mobile phone number; her email address is limited but her name is freely accessible. Finally, through the third rule, she overrides the rule related with her credit card number and provides this information freely to the *BuyTicket* (Figure 5) class of services.

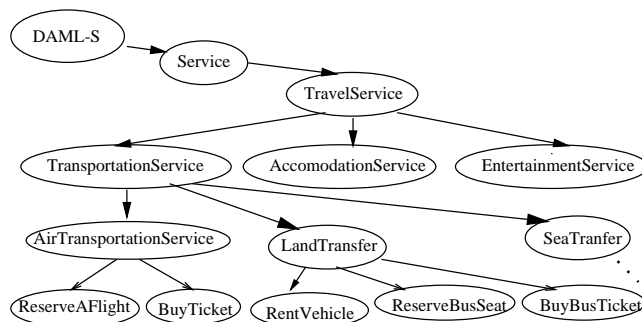
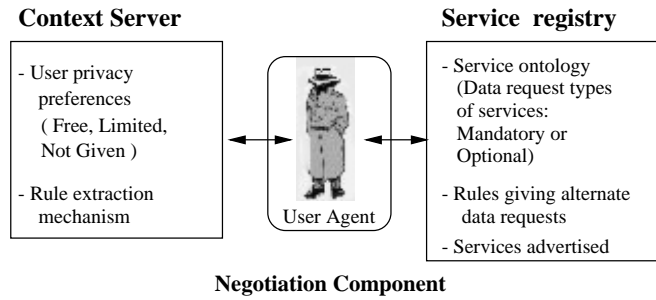


Fig. 5. An Example Class Hierarchy for Travel Domain



**Fig. 6.** General Architecture of the Framework

Different rules may impose conflicting permissions on data elements. When a conflict arises among rules associated with a given service, the final rule for the data element is determined based on the rule priorities. *NotGiven* rule dominates over other rules. *Free* rule has the least priority and *Limited* rule's priority is between these two. Among the rules associated with a data element, the one with the highest priority, i.e. the most restricted permission level, is chosen to be the rule for that element.

## 2.4 Architecture of the System

The general architecture of the system is shown in Figure 6. A context server, which is a trusted authority, stores the privacy preferences of a user based on domain specific service ontologies. The service registry, on the other hand, stores the advertised services, their semantic descriptions and the rules for alternate data requests by the Web Service. We propose the service semantics to be stored by conforming to the DAML-S upper ontology.

## 3 Negotiation Component

Negotiation is the set of activities where the user's data privacy preferences are compared with service's data request in order to reach an agreement. For this purpose, first the data requests of the Web service along with the rules defining alternatives are obtained from the service ontology. Then user's rules are compared with Web service's data requests. If an agreement can not be reached, the alternative data requests expressed through conditional statements provided by the service is used.

Rule extraction facility is provided by the context server, while the negotiation component is used by the user agent.

### 3.1 Extraction of Preference Rules

The initial activity of the rule extraction process is to obtain the Web service's data requests from the service ontology. In order to find the rules governing the

privacy of the data elements requested by the Web service, a temporary service graph is created. This graph is used for determining the privacy rules for those data elements that do not have any rule associated with themselves and need to inherit them from their parent nodes in the ontology.

As an example assume that the user wishes to invoke a *BuyTicket* service and the data requests for this service are a “name”, and a “credit card number”. Assume further that the user privacy rules at this level provide for “credit card number” but no rule is given for “name” as shown in Figure 4. The privacy rule for this data element should be obtained from the rules given at the higher levels of the temporary service graph.

The rule extraction process has two phases:

- 1st Phase: Upward Traversal
  - At each node, extract rules related with the needed data elements.
  - Request the rules from parents for undetermined data elements.
- 2nd Phase: Downward traversal
  - For each data element that is needed, receive the rule from the parents.
  - Based on the priorities determine the final rule.
  - Push rules downwards in the temporary service graph. Output the rules applicable to the data elements requested by the service.

If more than one rule is applicable to a data element, the final rule is determined from the rule priorities as mentioned in Section 2.3. The conflict resolution basically declares that the most restricted rule should always be chosen, e.g. *NotGiven* over *Limited* rule.

During the second phase, where the temporary service graph is traversed top-to-bottom, the rules extracted at each node are pushed to the children, and incorporated into the rules of child nodes. In this way the final rule set is collected at the node of the requestor service. When there are more than one parent service nodes, the final rule associated with an element is determined by the resolution process mentioned above, i.e. the most restricted permission level is chosen.

### 3.2 Negotiation of Data Elements

When the data provided by the user does not match with the data requested by the service, that is, when there is a mandatory data element requested by the Web service that is not given by the user, the alternative rules provided by the Web service (if any) are used to reach an agreement.

Negotiation process basically tries to find out if another data element can be used in place of a “not given” but “mandatory” data element. The aim is to determine the set of elements that can be exchanged between the parties, without violating user’s privacy.

## 4 An Example Scenario

In this section, we provide a scenario to better illustrate the concepts presented. Assume a domain specific service ontology for travel domain as given in Figure 5 and assume that the user wishes to invoke a service which is a member of *BuyTicket* class. Assume further that *BuyTicket* class of services require user's name, mobile phone number and credit card as mandatory data elements. User's age and email address are optional information for the service as shown in Figure 7.

The *BuyTicket* class of services further declare that if user's mobile phone number is not available then her email address is mandatory and her home phone is an optional data element. Recall that all this information is stored with the service ontology. We process this information to obtain the rules given in Figure 8.

```

<daml:Class rdf:ID="BuyTicket">
  <rdfs:subClassOf rdf:resource=
    "#AirTransportationService"/>
  </rdfs:subClassOf>
</daml:Class>
<rdf:Property rdf:ID="name">
  <rdfs:subPropertyOf rdf:resource="#mandatory"/>
  <rdfs:domain rdf:resource="#BuyTicket"/>
  <rdfs:range rdf:resource="&xsd:string"/>
</rdf:Property>
<rdf:Property rdf:ID="mobile.Phone">
  <rdfs:subPropertyOf rdf:resource="#mandatory"/>
  <rdfs:domain rdf:resource="#BuyTicket"/>
  <rdfs:range rdf:resource="&xsd:string"/>
</rdf:Property>
<rdf:Property rdf:ID="creditCardNo">
  <rdfs:subPropertyOf rdf:resource="#mandatory"/>
  <rdfs:domain rdf:resource="#BuyTicket"/>
  <rdfs:range rdf:resource="&xsd:string"/>
</rdf:Property>
<rdf:Property rdf:ID="age">
  <rdfs:subPropertyOf rdf:resource="#optional"/>
  <rdfs:domain rdf:resource="#BuyTicket"/>
  <rdfs:range rdf:resource="&xsd:string"/>
</rdf:Property>
<rdf:Property rdf:ID="emailAddress">
  <rdfs:subPropertyOf rdf:resource="#optional"/>
  <rdfs:domain rdf:resource="#BuyTicket"/>
  <rdfs:range rdf:resource="&xsd:string"/>
</rdf:Property>

```

Fig. 7. Input Parameters of *BuyTicket* Class

Note that the actual service instance may request other *interactive* parameters that are not directly related with the privacy of the user's context information. Such data may be received either directly from the user or through

```

<pri:Data rdf:ID="Data">
  <pri:Mandatory rdf:resource=
    ".../UserContextTaxonomy#Name"/>
  <pri:Optional rdf:resource=
    ".../UserContextTaxonomy#Age"/>
  <pri:Mandatory rdf:resource=
    ".../UserContextTaxonomy#Mobile.Phone"/>
  <pri:Optional rdf:resource=
    ".../UserContextTaxonomy#EmailAddress"/>
  <pri:Mandatory rdf:resource=
    ".../UserContextTaxonomy#CreditCardNo"/>
</pri:Data>
<pri:IfRule>
  <pri:If rdf:parseType="Resource">
    <pri:NotGiven rdf:resource=
      ".../UserContextTaxonomy#Mobile.Phone"/>
  </pri:If>
  <pri:Then rdf:parseType="Resource">
    <pri:Mandatory rdf:resource=
      ".../UserContextTaxonomy#EmailAddress"/>
    <pri:Optional rdf:resource=
      ".../UserContextTaxonomy#Home.Phone"/>
  </pri:Then>
</pri:IfRule>

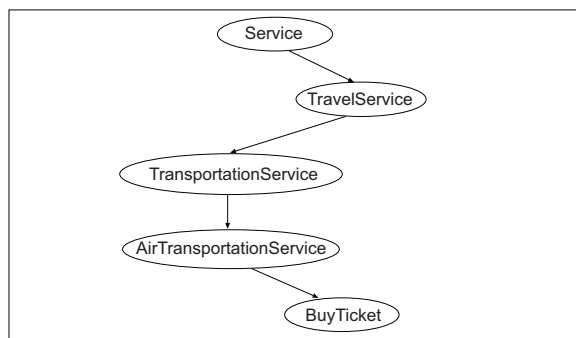
```

**Fig. 8.** *BuyTicket* Web service's Data Request

the user's agent. For example, for a ticket buying service, this information may include the flight destination and seat class.

#### 4.1 Rule Extraction

The initial activity of rule extraction is to generate a temporary service graph that contains the service node in question (*BuyTicket*) and all of its ancestors. Figure 9 presents temporary service graph for *BuyTicket* service class. The corresponding data request of the service is given in Figure 8.



**Fig. 9.** Temporary service graph generated for *BuyTicket* service.

In the first phase of the rule extraction process, the service ontology is queried to extract the input parameters and the alternate data request rules of *BuyTicket* service class. The user context rule segment associated with *BuyTicket*, declares that the user releases *CreditCardNo* freely to the services of this class as shown in Figure 4. The service is in need of further data elements and the upper levels of the temporary service graph is processed to obtain these data elements.

Figure 10 shows which higher level service classes provide the data elements requested by *BuyTicket* class as *Free* or *Limited* elements after completing the first phase of the process.

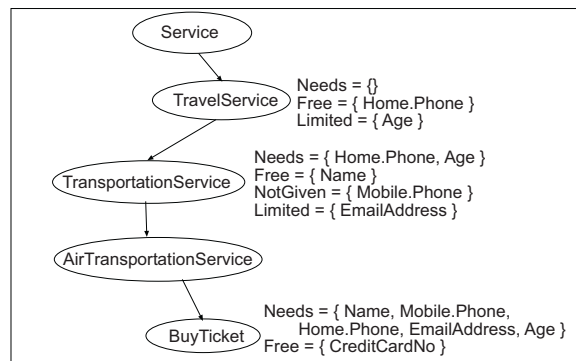


Fig. 10. State of temporary service graph, after Phase 1

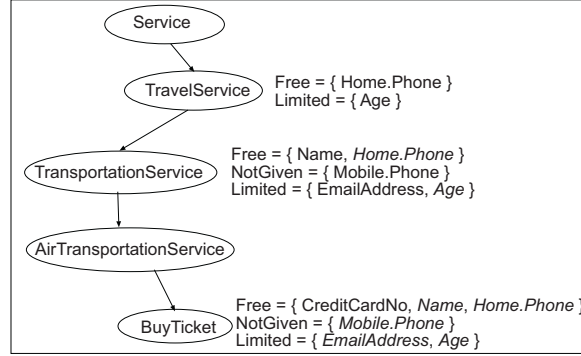
During the second phase of rule extraction process, the temporary service graph is traversed downwards starting from the *TravelService* node, while each service receives rules for the elements it needs from its parent service node.

Figure 11 presents the temporary service graph at the end of the second phase of rule extraction. The data elements shown in *italics>* are the ones inherited from parent services in the graph. The permission rules collected at *BuyTicket* service node define the rules associated with each element referred in the data request.

The negotiation process may start when the user's privacy preferences regarding the data elements, i.e. the permission levels, requested by the service are determined.

## 4.2 Negotiation

Following the running example, note that the mandatory *Mobile.Phone* is not given to the service. However, through the alternative rules (Figure 8), the service states that it accepts email address in place of mobile phone number. In addition, home phone number is also requested as an optional data element. Therefore, the conditional request is triggered, and the new alternative requests are added into the original input set. Note that, initially, user's email address was an optional



**Fig. 11.** State of temporary service graph, at the end of rule extraction process.

data element for the service. When the conditional statement is triggered, this item becomes mandatory in the input set.

The resulting input set is as follows:

$$\begin{array}{l}
 \text{Mandatory} = \{ \text{Name, CreditCardNo,} \\
 \quad \text{EmailAddress} \} \\
 \text{Optional} = \{ \text{Age, Home.Phone} \}
 \end{array}$$

The permission levels of data elements obtained from the rule extraction process is as follows:

$$\begin{array}{l}
 \text{Free} = \{ \text{Name, Home.Phone, CreditCardNo} \} \\
 \text{Limited} = \{ \text{Age, EmailAddress} \} \\
 \text{NotGiven} = \{ \text{Mobile.Phone} \}
 \end{array}$$

In the final phase of the negotiation activities, the necessity levels for the requested data elements are compared with the permission levels extracted from user's data privacy preferences. The mandatory data elements **Name** and **CreditCardNo** are provided with *Free* rule, hence the user agrees to provide these data elements. The mandatory data element **EmailAddress** is associated with *Limited* rule in the privacy preferences of the user. As this element is crucial for the service enactment, it is also released by the user.

The home phone number of the user is provided freely, independent of the necessity level. Hence, it is included in the agreement set. However, the age of the user is not presented to the service, because the privacy preferences states that this element is provided in a limited fashion. Recall that, elements that are associated with *Limited* rules in the privacy preferences, are released only if they are requested with *Mandatory* necessity level.

Even if the data element **Age**, is not released to the service, it is not a mandatory element for the service, hence does not hinder the service enactment. Therefore, there are no conflicts between service's input request and user's privacy preferences. An agreement is settled between the parties. The agreed-upon data

set, which determines the elements that may be passed to the service, is presented in the following:

$$\begin{array}{l} \text{Mandatory} = \{ \text{Name, CreditCardNo, EmailAddress} \} \\ \text{Optional} = \{ \text{Home.Phone} \} \end{array}$$

## 5 Related Work

Authentication services like Microsoft Passport [19] and AOL Screen Name [2], store and manage personal user data and provide single sign-in identity at different sites and pass personal information more easily. The stored personal data is generally limited to user identification and user contact information that can be used in basic e-commerce sessions.

[13] describes the ePerson project developed at the HP Labs. An ePerson is a personal representative on the net that is trusted by a user to store and make available personal information under appropriate controls. Such personal information includes user profiles, shared content and shared meta-data (such as annotations, comments, ratings and categorisations). However how privacy issues are handled in ePerson is not available in the literature.

[18] defines a vocabulary for composing policies to allow or deny access to the personal information that a policy governs. The work also describes how policies can be merged using negotiation rules and how Semantic Web logic processors reason through policies.

Among several approaches for privacy management using service policies and privacy preferences, the most mature one is the Platform for Privacy Preferences Project (P3P) [6] developed by the World Wide Web Consortium (W3C). P3P provides a simple, automated way for users to gain more control over the use of personal information on Web sites they visit. At its most basic level, P3P is a standardized set of multiple-choice questions, covering all the major aspects of a Web site's privacy policies. Taken together, they present a clear snapshot of how a site handles personal information about its users. P3P-enabled Web sites make this information available in a standard, machine-readable format. P3P enabled browsers can "read" this snapshot automatically and compare it to the consumer's own set of privacy preferences. P3P enhances user control by putting privacy policies where users can find them, in a form users can understand, and, most importantly, enables users to act on what they see.

The P3P Specification 1.0 [6] includes the definition of the syntax and semantics of a vocabulary to describe data uses, data recipients, data retention policy and other privacy disclosures in P3P privacy policy files. A base data schema defines a standard set of data elements that will be referenced from these policies, as well as a mechanism for associating policies with Web resources.

P3P policies are written in machine readable XML [26] format, facilitating XML Namespaces [27] to refer to the P3P policy vocabulary elements. The privacy vocabulary elements and the base data schema are defined using XML Schema [24, 25]. P3P policies declare the data elements that will be collected by the Web site and explain how the data will be used. They also identify the

recipients of the collected data, and make a variety of other disclosures including information about dispute resolution procedures and remedy processes. Textual description about the company and related contact information, may also be included in the policy files.

APPEL (A P3P Preference Exchange Language) [5] provides a standard way of defining the user privacy preferences in a set of preference rules, which can be used by the user agent to make automated and semi-automated decisions regarding the acceptance of privacy policies from P3P enabled Web sites. While the user agent may present the user preferences in some internal format, APPEL provides a standard way to do this.

Recently, the Web Services Architecture (WSA) Working Group at W3C which is tasked with producing a Web service reference architecture, has produced the critical success factors and requirements for the privacy of Web services [29]:

- The WSA must enable privacy policy statements to be expressed about Web services.
- Advertised Web service privacy policies must be expressed in P3P.
- The WSA must enable a consumer to access a Web service’s advertised privacy policy statement.
- The WSA must enable delegation and propagation of privacy policy.
- Web Services must not be precluded from supporting interactions where one or more parties of the interaction are anonymous.

These are the very basic requirements to enable privacy protection for the consumer of a Web service across multiple domains and services. We extend this basic architecture by exploiting the semantics of Web services. It should be noted that P3P does not intend to exploit Web semantics.

In fact only a few recent work address semantic issues for privacy management: [17] points out that a standard method of exchanging privacy policies, that is a privacy ontology, is needed for the Semantic Web.

## 6 Conclusions and Future Work

Privacy preferences of a user, define the rules that control the read access for personal information. In related specifications like P3P, privacy preferences are based on URLs’ of Web sites, as these technologies are mostly intended for Web browsing applications and interactive e-commerce sessions.

In this work, a privacy framework for Web services is proposed. Declaring privacy preferences on the basis of a service ontology prevents the user from repetitive specifications since the privacy preferences at the upper classes of the ontology are inherited by lower classes. Furthermore the framework presented allows Web services to declare alternate data requests if a mandatory input is not given by the user. In this way it becomes possible to automate the negotiation process with a Web service to reach an agreement.

The features provided within the proposed privacy scheme, i.e. service-class-based preferences, multi-level element types, consent-based policies and negotiation activities, are utilized to enhance the process of decision making in agent programs. What distinguishes our work is that in this privacy framework, domain specific Web service ontologies are used. How service ontologies can be stored into service registries and how service semantics can be related with the services advertised are available from our accompanying work [8–10].

There are a number of issues left as a future work:

- Privacy preferences should also include user’s choice of accepted data-use practices, such as the data retention policies of the service.
- What Web services need to know is not only user preferences but a “user context” that includes any information that can be used to characterize the user and her situation. Hence user context should include user’s local data obtained through sensors as well as any data stored about the user such as those stored in customer relationship management (CRM) systems to make effective use of Web services.
- This context information should be available to any authorized agent at any time, anywhere in a secure manner: This necessitates developing secure “context servers”, that is, the user context information should be available anywhere, any time to a variety of devices from desktops to mobile devices. Since these devices accept input in different mark up languages; the context servers need to recognize the device and provide the information in the format that can be accepted by the device. Note that if the user permits, information on user activities should be collected to further improve user context.
- More importantly, user context should be available in a format that is machine processable and interoperable. In this respect developing a user context ontology is essential.
- Yet all this will make privacy a graver concern for users. There is a need for trusted authorities for delivering user context to authorized requestors in a secure manner.

## References

1. Amazon.com Web services, <http://www.amazon.com/gp/browse.html/002--8640824-9000064?node=3435361>
2. AOL Screen Name, <http://my.screenname.aol.com>
3. Bargh, M. S., van Eijk, Ebben, P., Salden, A. H., “Agent-based Privacy Enforcement of Mobile services”, in Proc. of SSGRR Conference, Italy, January 2003.
4. Carey, M., Blevins, M., Takacs-Nagy, P., “Integration, Web Services Style”, IEEE Data Engineering Bulletin, Vol. 25, No. 4, December 2002.
5. Cranor L., Langheinrich M., and Marchiori M., *A P3P Preference Exchange Language 1.0 (APPEL 1.0)*, W3C Working Draft, [www.w3.org/TR/P3P-preferences](http://www.w3.org/TR/P3P-preferences), April 15, 2002
6. Cranor L., Langheinrich M., Marchiori M., Presler-Marshall M., Reagle J., *The Platform for Privacy Preferences 1.0 (P3P1.0) Specification*, W3C Recommendation, [www.w3.org/TR/P3P](http://www.w3.org/TR/P3P), April 16, 2002.

7. DAML Services Coalition (A. Ankolekar, M. Burstein, J. Hobbs, O. Lassila, D. Martin, S. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, H. Zeng), DAML-S: Semantic Markup for Web Services, in Proceedings of the International Semantic Web Working Symposium (SWWS), July 2001.
8. Dogac, A., Laleci, G., Kabak, Y., Cingil, I., "Exploiting Web Service Semantics: Taxonomies vs. Ontologies", IEEE Data Engineering Bulletin, Vol. 25, No. 4, December 2002, <http://www.research.microsoft.com/research/db/debull/issues-list.htm>.
9. Dogac, A., Cingil, I., Laleci, G. B., Kabak, Y., "Improving the Functionality of UDDI Registries through Web Service Semantics", 3rd VLDB Workshop on Technologies for E-Services (TES-02), Hong Kong, China, August 23-24, 2002.
10. Dogac, A., Kabak, Y., Laleci, G., "Enriching ebXML Registries with OWL Ontologies for Efficient Service Discovery", 14th Intl. Workshop on Research Issues on Data Engineering, Boston, USA, March 2004.
11. Dogac, A., Kabak Y., Laleci, G. , Sinir S., Yildiz A., Kirbas S., Gurcan Y., "Semantically Enriched Web Services for Travel Industry", Technical Report, METU-SRDC, December 2003, submitted for publication.
12. ebXML, <http://www.ebxml.org/>
13. e-person: Personal Information Infrastructure, <http://www.hpl.hp.com/semweb/e-person.htm>
14. Google Web Service API, <http://www.google.com/apis/>
15. Harmonise Project, IST-2000-29329, Tourism Harmonisation Network, <http://www.harmonise.org/>
16. Karjoth, G., Schunter, M., *A Privacy Model for Enterprises*, 15th IEEE Computer Security Foundations Workshop, June 24-26, 2002.
17. Kim, A., Hoffman, L. J., Martin, C. D., "Building Privacy into the Semantic Web: An Ontology Needed Now", in Proc. of Semantic Web Workshop, Hawaii, USA, 2002.
18. Lee, R., "Personal Data Protection in the Semantic Web", ME Thesis, MIT, USA, 2002, <http://www.w3.org/2002/01/pedal/thesis.html>
19. Microsoft Passport, <http://www.microsoft.com/myservices/passport>.
20. IST-1-002104-STP Satine Project, <http://www.srdc.metu.edu.tr/webpage/projects/satine>
21. Open Travel Alliance (OTA), <http://www.opentravel.org/>
22. Simple Object Access Protocol (SOAP), <http://www.w3.org/TR/SOAP/>
23. Universal Description, Discovery and Integration (UDDI), [www.uddi.org](http://www.uddi.org)
24. Thompson, H. S., Beech, D., Maloney, M., and Mendelsohn, N., *XML Schema Part 1: Structures*, W3C Recommendation, [www.w3.org/TR/xmlschema-1](http://www.w3.org/TR/xmlschema-1), May 2, 2001.
25. Biron, P., Malhotra, A., *XML Schema Part 2: Datatypes*, W3C Recommendation, [www.w3.org/TR/xmlschema-2](http://www.w3.org/TR/xmlschema-2), May 2, 2001.
26. Bray, T., Paoli, J., Sperberg-McQueen, C.M., Maler, E., *Extensible Markup Language (XML) 1.0 (Second Edition)*, W3C Recommendation, [www.w3.org/TR/REC-xml](http://www.w3.org/TR/REC-xml), October 6, 2002.
27. Bray, T., Hollander, D., Layman, A., *Namespaces in XML*, W3C Recommendation, [www.w3.org/TR/REC-xml-names](http://www.w3.org/TR/REC-xml-names), January 14, 1999.
28. Wroe, C., Stevens, R., Goble, C., Roberts. A., Greenwood, M., "A Suite of DAML+OIL Ontologies to Describe Bioinformatics Web Services and Data", Intl. Journal of Cooperative Information Systems, to appear.
29. Web Services Architecture Requirements, <http://www.w3.org/TR/2004/NOTE-wsa-reqs-20040211/>

18 Arif Tumer, Asuman Dogac, and I. Hakki Toroslu

30. Web Service Description Language (WSDL), <http://www.w3.org/TR/wsdl>