A Semantic based Privacy Framework for Web Services

Arif Tumer, Asuman Dogac, I. Hakki Toroslu
Software Research and Development Center &
Dept. of Computer Eng.
Middle East Technical University (METU)
06531 Ankara Türkiye
email: arif,asuman@srdc.metu.edu.tr, toroslu@ceng.metu.edu.tr

Abstract

There are some key considerations in developing a privacy mechanism such as revealing only the minimal pertinent information about the user, not to overwhelm the users while declaring their privacy preferences and requiring no or only limited user interaction.

In this paper, with these considerations in mind, 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. We propose that 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

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 [6] 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 [18].

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 the semantics can be exploited is for 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 over-

whelm the users while declaring their privacy preferences. Indeed declaring privacy prefences 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 [5] 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 services to declare their input parameters as *Mandatory* or *Optional*. We show how DAML-S Service Profile input parameter specification [6] 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 4). 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: first, the privacy preferences are declared for a group of services. Furthermore, a user may declare the same policy for several different service groups. The effort required by the user is further minimized since the privacy preferences at the upper level classes are inherited by lower level service classes. Note that a user can override a privacy preference at any level she likes. Secondly, 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 paper is organized as follows: Section 2 describes the related work. In Section 3, a privacy framework for Web services based on domain specific service ontologies is presented. Section 4 describes the negotiation process between the user agent and the Web service. In Section 5, an example scenario is given to illustrate the concepts. Finally, Section 6 concludes the paper.

2 Related Work

Authentication services like Microsoft Passport [15] and AOL Screen Name [1], store and manage personal user data and provide single sing-in identity in 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.

[11] describes the ePerson project developed at the HP Labs. An ePerson is a personal rep-

resentative 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.

Among several approaches for privacy management using service policies and privacy preferences, the most mature one is the Platform for Privacy Preferences Project (P3P) [5] developed by the World Wide Web Consortium (W3C). P3P enables Web sites to express their privacy practices in a standard format that can be retrieved automatically and interpreted easily by user agents like Web browsers.

The P3P Specification 1.0 [5] 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.

APPEL (A P3P Preference Exchange Language) [4] 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.

It should be noted that P3P is for Web sites and does not intend to exploit Web semantics. In fact only a few recent work address semantic issues for privacy management: [13] points out that a standard method of exchanging privacy policies, that is a privacy ontology, is needed for the Semantic Web.

[14] 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.

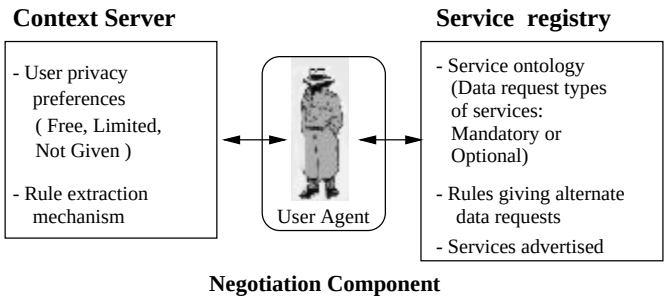

Figure 1: General Architecture of the Framework

What distinguishes our work is that we propose a privacy framework for Web services based on domain specific Web service ontologies. 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 [7, 8, 9].

3 A Privacy Framework for Web Services

The general architecture of the system is shown in Figure 1. 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.

There are two basic elements in the privacy model, the data requests of Web services and the privacy preferences of the users.

3.1 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 the 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 requesting 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.

```
<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>
```

Figure 2: Defining the types of Data Element Requests through DAML-S

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 [6].

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.

As an example, consider the conditional statements given in the following:

```
<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>
```

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.

3.2 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 3.

```
<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>
```

Figure 3: User Context Privacy Rules

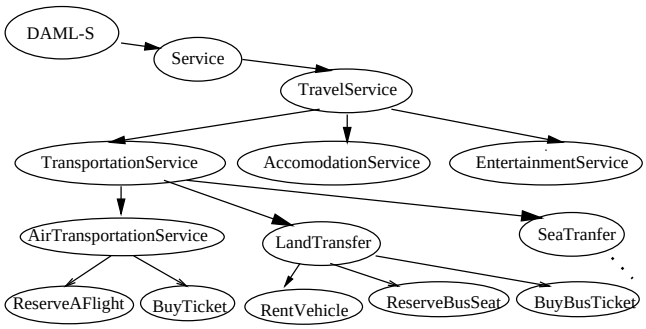

Figure 4: An Example Class Hierarchy for Travel Domain

The first rule is associated with the top level service class defined for the travel ontology of Figure 4. 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. Then through the third rule, she overrides the rule related with her credit card number and provides this information freely to the *BuyTicket* (Figure 4) class of services.

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.

4 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.

4.1 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 parents.

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 3. 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 is 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 3.2. 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.

4.2 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.

5 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 4 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 5.

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 6.

Note that the actual service instance may request other *interactive* parameters that are not directly related with the privacy of the user's

```
<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:resorce="#BuyTicket"/>
   <rdfs:range rdf:resource="&xsd;string"/>
</rdf:Property>
<rdf:Property rdf:ID="mobile.Phone">
   <rdfs:subPropertyOf rdf:resource="#mandatory"/>
   <rdfs:domain rdf:resorce="#BuyTicket"/>
   <rdfs:range rdf:resource="&xsd;string"/>
</rdf:Property>
<rdf:Property rdf:ID="creditCardNo">
   <rdfs:subPropertyOf rdf:resource="#mandatory"/>
   <rdfs:domain rdf:resorce="#BuyTicket"/>
   <rdfs:range rdf:resource="&xsd;string"/>
</rdf:Property>
<rdf:Property rdf:ID="age">
   <rdfs:subPropertyOf rdf:resource="#optional"/>
   <rdfs:domain rdf:resorce="#BuyTicket"/>
   <rdfs:range rdf:resource="&xsd;string"/>
</rdf:Property>
<rdf:Property rdf:ID="emailAddress">
   <rdfs:subPropertyOf rdf:resource="#optional"/>
   <rdfs:domain rdf:resorce="#BuyTicket"/>
   <rdfs:range rdf:resource="&xsd;string"/>
</rdf:Property>
```

Figure 5: Input Parameters of *BuyTicket* Class

```
<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>
```

Figure 6: *BuyTicket* Web service's Data Request

context information. Such data may be received either directly from the user or through the user's agent.

5.1 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 7 presents temporary service graph for *BuyTicket* service class. The corresponding data request of the service is given in Figure 6.

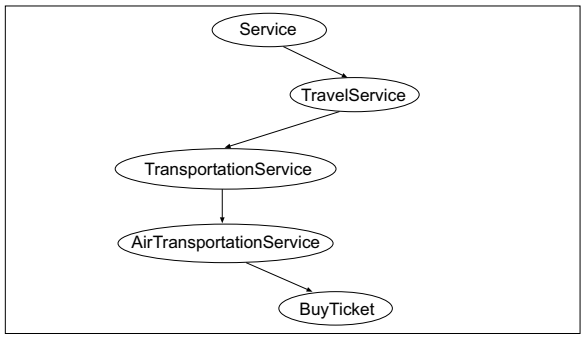

Figure 7: Temporary service graph generated for BuyTicket service.

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 3. 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 8 shows which higher level services provide the data elements requested by *BuyTicket* class of services as *Free* or *Limited* elements after completing the first phase of the process.

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

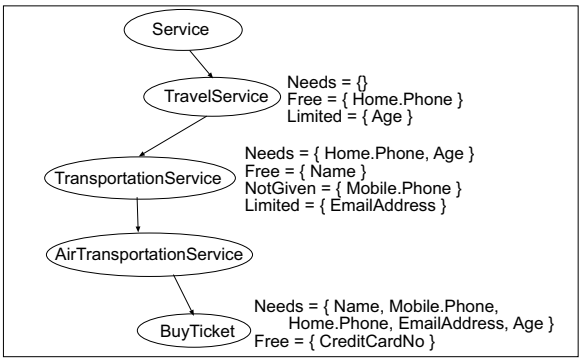

Figure 8: State of temporary service graph, after Phase 1

parent service node.

Figure 9 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.

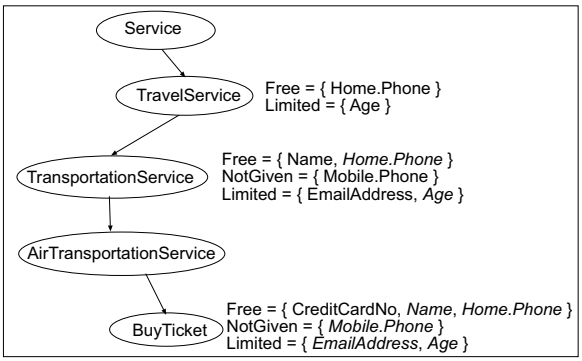

Figure 9: State of temporary service graph, at the end of rule extraction process.

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

5.2 Negotiation

Following the running example, note that the mandatory `Mobile.Phone` is not given to the service. However, through the alternative rules (Figure 6), 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 data element for the service. When the conditional statement is triggered, this item is made mandatory in the input set.

The resulting input set is as follows:

Mandatory	=	{ `Name`, `CreditCardNo`, `EmailAddress` }
Optional	=	`Age`, `Home.Phone`

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

Free	=	{ `Name`, `Home.Phone`, `CreditCardNo` }
Limited	=	{ `Age`, `EmailAddress` }
NotGiven	=	{ `Mobile.Phone` }

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 determine the elements that may be passed to the service, is presented in the following:

Mandatory	=	{ `Name`, `CreditCardNo`, `EmailAddress` }
Optional	=	`Home.Phone`

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 are inherited by lower classes. Furthermore the presented framework 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.

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, any where in a secure manner: This necessitates developing globally accessible, secure "context servers", that is, the user context information should be available any where, any time to a variety of devices from desktops to mobile devices. Since these devices accept input in different mark up languages; the context server needs 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] AOL Screen Name, http://my.-screenname.aol.com

[2] M. S. Bargh, R. van Eijk, P. Ebben, A. H. Salden, "Agent-based Privacy Enforcement of Mobile sevices", in Proc. of SSGRR Conference, Italy, January 2003.

[3] Carey, M., Blevins, M., Takacsi-Nagy, P., "Integration, Web Services Style", IEEE Data Engineering Bulletin, Vol. 25, No. 4, December 2002.

[4] 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`, April 15, 2002

[5] 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`, April 16, 2002.

[6] 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.

[7] 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.

[8] 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.

[9] Dogac, A., Kabak, Y., Laleci, G., "Exploiting Web Service Semantics through ebXML Registries and Software Agents", submitted for publication.

[10] ebXML, http://www.ebxml.org/

[11] e-person: Personal Information Infrastructure, http://www.hpl.hp.com/-semweb/e-person.htm

[12] Karjoth G., Schunter M., *A Privacy Model for Enterprises*, 15th IEEE Computer Security Foundations Workshop, June 24-26, 2002.

[13] A. Kim, L. J. Hoffman, C. D. Martin, "Building Privacy into the Semantic Web: An Ontology Needed Now", in Proc. of Semantic Web Workshop, Hawaii, USA, 2002.

[14] R. Lee, "Personal Data Protection in the Semantic Web", ME Thesis, MIT, USA, 2002, http://www.w3.org/2002/-01/pedal/thesis.html

[15] Microsoft Passport, http://www.-microsoft.com/myservices/passport.

[16] Simple Object Access Protocol (SOAP), http://www.w3.org/TR/SOAP/

[17] Universal Description, Discovery and Integration (UDDI), www.uddi.org

[18] 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.

[19] Web Service Description Language (WSDL), http://www.w3.org/TR/wsdl