

DESIGN AND IMPLEMENTATION OF A PRIVACY FRAMEWORK
FOR WEB SERVICES IN THE TRAVEL DOMAIN

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MEHMET ERKANAR

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

NOVEMBER 2005

Approval of the Graduate School of Natural and Applied Sciences.

Prof. Dr. Canan Özgen
Director

I certified that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Ayşe Kiper
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Prof. Dr. Asuman Doğaç
Supervisor

Examining Committee Members

Prof. Dr. Asuman Doğaç (CENG, METU) _____

Prof. Dr. Volkan Atalay (CENG, METU) _____

Prof. Dr. Özgür Ulusoy (CS, BİLKENT) _____

Assoc. Prof. Dr. İ. Hakkı Toroslu (CENG, METU) _____

Assoc. Prof. Dr. Nihan K. Çiçekli (CENG, METU) _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Mehmet Erkanar

Signature :

ABSTRACT

DESIGN AND IMPLEMENTATION OF A PRIVACY FRAMEWORK FOR WEB SERVICES IN THE TRAVEL DOMAIN

Erkanar, Mehmet

M. S., Department of Computer Engineering

Supervisor: Prof. Dr. Asuman Doğaç

November 2005, 90 pages

A web service is a collection of functions that are packaged as a single entity and published to the network for use by other programs. Web services are building blocks for creating open distributed systems, and allow companies and individuals to quickly and cheaply make their digital assets available worldwide. With considerable interoperability, privacy management becomes an inevitable concern of the web services. Companies and individuals should be able to restrict the information available about themselves and specify the use of that information in order to protect their confidentiality.

In the thesis, a privacy framework has been designed and implemented in order to prepare and match privacy documents for web services. Privacy documents are prepared based upon message ontologies which describe the input data of web services. Service requestors and providers prepare their own privacy documents which are going to be checked before the web service transaction begins. Privacy content has been derived from the World Wide Web Consortium's Platform for Privacy Preferences specification.

Keywords: E-Business, Web Services, Web Service Privacy, Ontologies, Privacy Matching.

ÖZ

TURİZM ALANINDA AĞ SERVİSLERİ İÇİN GİZLİLİK ALT YAPISI TASARIM VE GELİŞTİRMESİ

Erkanar, Mehmet

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Danışman: Prof. Asuman Doğaç

Kasım 2005, 90 sayfa

Ağ servisleri, diğer programlar tarafından kullanılmaya hazır, bir araya getirilerek ağda kullanıma açılmış fonksiyonlar bütünüdür. Aynı zamanda, ağ servisleri, açık dağıtık sistemler yaratmanın da yapı taşlarıdır. Kişilerin ve şirketlerin, varlıklarını tüm dünyaya açmalarını sağlarlar. Ağ servislerinde var olan birlikte çalışırlık yüzünden kişisel gizlilik yönetimi, kaçınılmaz öneme sahip olmuştur. Şirketler ve bireyler, kendilerine ait açık bilgilerini sınırlayabilmeli, ve bu bilgilerin kullanımını belirleyebilmelidir.

Tezde, bir kişisel gizlilik alt yapısı tasarlanmış ve geliştirilmiştir. Bu alt yapı, ağ servislerine ait olan gizlilik belirleme belgelerinin hazırlanmasını ve karşılaştırılmasını sağlamaktadır. Gizlilik belirleme belgeleri, ağ servislerinin girdi ve çıktılarının tanımlandığı mesaj ontolojileri üzerine kurulmaktadır. Servis isteyenler ve verenler, servis işlemleri başlamadan önce karşılaştırılacak olan, kendilerine ait belgeleri oluşturabilmektedir. Gizlilik belgeleri içeriği, W3C oluşumunun Gizlilik Tercihleri Platformu belirtiminden alınmıştır.

Anahtar Kelimeler: E-Ticaret, Ağ Servisleri, Ağ Servisleri Gizliliği, Ontolojiler, Gizlilik Karşılaştırması.

To My Wife...

For being always with me...

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Prof. Dr. Asuman Doğaç, for his guidance and encouragement throughout the research. To my family, I offer sincere thanks for their emotional support.

TABLE OF CONTENTS

ABSTRACT.....	iv
ÖZ.....	v
ACKNOWLEDGEMENTS.....	vii
TABLE OF CONTENTS.....	viii
LIST OF TABLES.....	xi
LIST OF FIGURES.....	xii
CHAPTER	
1. INTRODUCTION.....	1
1.1. Organization of the Thesis.....	2
2. ENABLING TECHNOLOGIES.....	3
2.1. Web Services.....	3
2.1.1. SOAP (Simple Object Access Protocol).....	5
2.1.2. UDDI (Universal Description, Discovery and Integration Service).....	5
2.1.3. WSDL (Web Services Definition Language).....	6
2.2. E - Business and Web Services.....	9
2.3. Web Service Standards.....	11
2.3.1. Core Standards.....	12
2.3.2. Other Standards.....	13
2.4. Privacy Protection.....	15
2.4.1. A Privacy Model For Web Services.....	16
2.4.2. Privacy Protection Tools.....	17
2.5. Ontology Concept.....	17
2.5.1. OWL.....	18

2.5.2.	JENA Library	19
2.6.	Platform for Privacy Preferences Project (P3P).....	19
2.7.	Policy Management	30
2.7.1.	Policy Assertions	30
2.7.2.	Policy Expressions.....	31
2.7.3.	Policy Operators	31
2.7.4.	Other Policy Implementations	32
2.8.	SATINE Project.....	33
2.8.1.	The Open Travel Alliance (OTA).....	34
3.	DESIGN OF THE WEB SERVICE PRIVACY FRAMEWORK.....	36
3.1.	Use Cases.....	36
3.2.	Sequence Diagrams.....	37
3.2.1.	Service Provider Creates a Privacy Document Sequence.....	37
3.2.2.	Service Requestor Creates a Privacy Document Sequence	38
3.2.3.	Service Provider Modifies an Existing Privacy Document	39
3.2.4.	Service Requestor Modifies an Existing Privacy Document.....	40
3.2.5.	Compare Privacy Documents	40
3.3.	Class Diagrams	41
4.	IMPLEMENTATION OF THE WEB SERVICE PRIVACY FRAMEWORK	42
4.1.	Work Done.....	42
4.2.	Detailed Explanation.....	43
4.2.1.	Usage	43
4.2.2.	Privacy Policy Management	44
4.2.3.	Preparing Documents	45
4.2.4.	Matching Documents.....	51
5.	CONCLUSION	58
5.1.	Work Realized	58
5.2.	Related Work	59

5.3. Comments	59
5.4. Future Work.....	60
REFERENCES	61
APPENDIX A.....	65

LIST OF TABLES

TABLE

1.	UDDI.....	Error! Bookmark not defined.
2.	Usage Values	31
3.	Data Policy Attributes and System Behaviors	45
4.	Some statement template comparisons	57

LIST OF FIGURES

FIGURES

1.	Generic Simple web service architecture [30]	4
2.	Web Service Architecture	4
3.	WSDL Components	7
4.	HelloWorld.wsdl	9
5.	Core Web Service Standards	12
6.	Web Services Specification Stack [32]	15
7.	Privacy Protection Tools	17
8.	Sample Policy	30
9.	Create and modify use case diagrams	37
10.	Matching use case diagram	37
11.	Service provider creates a privacy document sequence diagram	38
12.	Service requestor creates a privacy document sequence diagram	39
13.	Service provider modifies an existing privacy document sequence diagram	40
14.	Privacy matching sequence diagram	41
15.	Privacy Document Usage	43
16.	Constructing an ontology tree	46
17.	Ontology Tree	47
18.	Dispute frame	48
19.	Statement template frame	49
20.	Disputes that match	52
21.	Disputes that do not match	52
22.	Ontology Trees	53
23.	Ontology Trees 2	54
24.	Trees with statement templates	56

25.	Packages.....	65
26.	PrivacyTool Class	66
27.	AccessPanel Class.....	67
28.	ComparisonFrame Class	68
29.	DataPanel Class	70
30.	Dispute package	71
31.	MainFrame Class	74
32.	PrivacyFrame Class	75
33.	PnlPurpose Class.....	77
34.	PurposeTable Class.....	78
35.	PurposeTableModel Class.....	78
36.	ConsequencePanel Class.....	79
37.	RecipientPanel Class.....	79
38.	RecipientTable Class	80
39.	RecipientTableModel Class	81
40.	gui.remedy package	81
41.	RetentionPanel class	83
42.	gui.rule package	84
43.	gui.statement package and relationships with other panels.....	86
44.	XMLFrame class.....	89
45.	XMLDocumentCreator class	89
46.	PrivacyParser class.....	90

CHAPTER 1

1. INTRODUCTION

A web service is a self-describing, self-contained, modular unit of application logic that provides some business functionality to other applications through an Internet connection. Applications access web services via ubiquitous web protocols and data formats with no need to worry about how each web service is implemented. Web services can be mixed and matched with other web services to execute a larger workflow or business transaction. [3]

With the emergence of those web services all around the world, e-business will become easier. Business structures will be composed of more simple atomic parts using the interoperable nature of web services. On the other side, e-business transactions require security principles such as confidentiality and trust, so web service security standards are constantly growing and being implemented. Considerable work, including from the main firms of the computer world, is dedicated to the web service security standards. Those standards enclose several specific enhancements such as web service policy, trust, authorization and privacy.

Privacy is a necessary concern in electronic commerce. It is almost impossible to complete a transaction without revealing some personal data – an electronic mail address, credit card number, or product preference [8]. Users may be unwilling to provide this necessary information or even to browse online if they believe their privacy is invaded or threatened. Fortunately, there are technologies to help users protect their privacy against threats.

The goal of the thesis is to provide a mechanism to describe privacy constraints in order to be used in web service transactions. Privacy content applied to the data has been obtained from the World Wide Web Consortium's P3P (Platform for Privacy Preferences Project) specification [2]. P3P has been used for web browsers to handle privacy issues between web servers. In this work, a generic approach has been realized; P3P documents have been prepared separately in eXtensible Markup Language (XML) format, so those documents can be embedded to any XML document. Also in order to enrich privacy, a comprehensive web service privacy policy has been developed based on the work described in [1]. In that work, new policy elements for service providers and requestors, new rules that can enable users to define complex policies, and semantic relationships amongst those elements have been described. Finally, privacy documents are built upon message ontologies, i.e. data whose privacies will be set are extracted from message ontologies. Using ontologies, the writer tried to make a connection to semantic web [3].

1.1. Organization of the Thesis

The focus of the thesis is to design and implement a utility software to prepare web service privacy documents based on domain ontologies. Beyond this introductory chapter, the thesis is organized as follows: In Chapter 2, the background information on web services, web service standards, web service privacy, ontologies are included. Chapter 3 describes the electronic business transactions and the use of web service privacy principles in detail. In chapter 4, interface layer that is developed to build the web service privacy documents is presented. Chapter 5 concludes, and presents work that has been conducted in the thesis and further work. Framework design is modeled with Unified Modeling Language (UML). Documentation for the design is represented in Appendix A.

CHAPTER 2

2. ENABLING TECHNOLOGIES

2.1. Web Services

Web services are automated software that can be initiated over the internet by another software packet. Web services are a fairly new branch of web applications. They are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web [17]. Web services are developed to perform actions, which can be any task from simple requests to complicated business processes. Once a web service is deployed in a proper place, other applications including web services can discover and invoke the deployed service.

Although there are already middleware platforms such as RMI, Jini, CORBA, DCOM, web services gain more importance. Those middleware platforms cannot provide adequate interoperability and uniformity, which web services can provide. Web as an information distributor, with its simplicity of access and ubiquity, are important in resolving the fragmented middleware world where interoperability is hard to come by. The Web complements these platforms by providing a uniform and widely accessible interface and access glue over services that are more efficiently implemented in a traditional middleware platform.

A web service is a layer for programmatic access to a service, which is implemented by other kinds of middleware. Access consists of request handling (a listener) and a facade that exposes the operations supported by the business logic [30]. The logic is implemented by a traditional middleware platform.

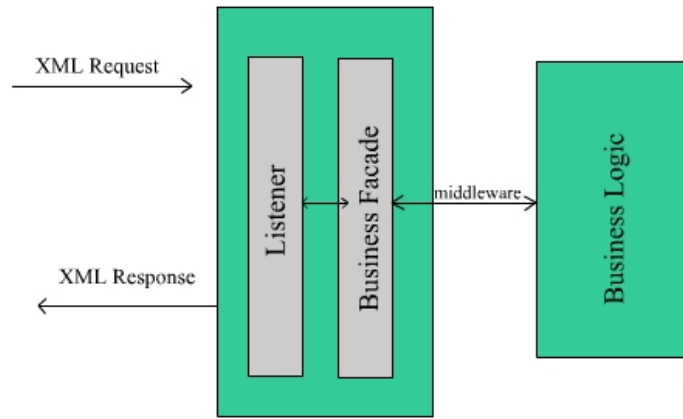


Figure 2.1 Generic Simple web service architecture [30]

The basic platform of web services is XML. XML provides a meta-language in which specialized languages can be written to express complex interactions between clients and services or between components of a composite service. Behind the facade of a web server, the XML request message is converted to a middleware request and the results are converted back to XML format.

Alongside XML, there are several support elements of web services. These elements are served to be used in discovery, security, transactions and authentication of web services. They also cause web to be a more functional platform. SOAP, UDDI, WSDL are amongst those support elements.

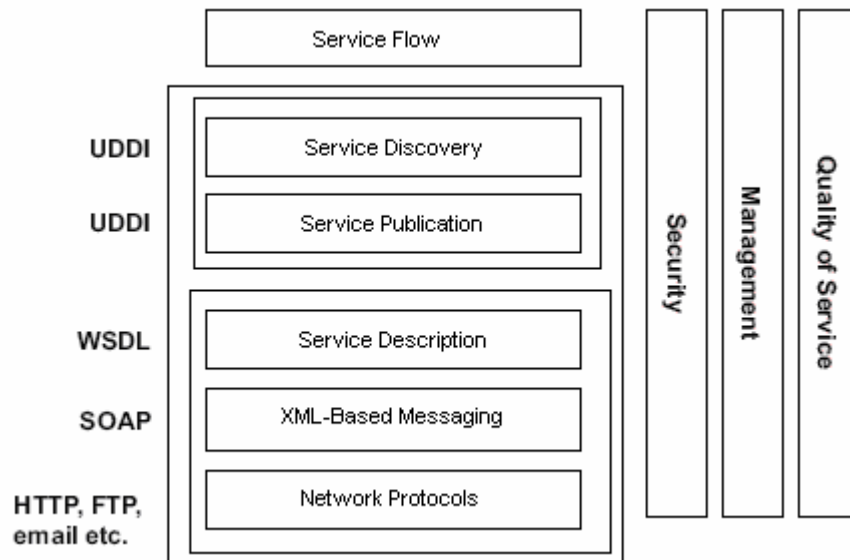


Figure 2.2 Web Service Architecture

2.1.1. SOAP (Simple Object Access Protocol)

SOAP is used for remote invocation. SOAP is a protocol specification that defines a uniform way of passing XML-encoded data. It also defines a way to perform remote procedure calls using HTTP as the underlying communication protocol.

SOAP provides interoperability between web services even middleware components of those services completely differ. Architecturally, sending messages as plain XML has advantages in terms of ensuring interoperability. The middleware firms seem willing to tolerate the costs of parsing and serializing XML in order to scale their approach to wider networks.

SOAP was submitted in 2000 to the W3C as a Note by IBM, Microsoft, UserLand, and DevelopMentor first. The further development of SOAP is now in the care of the W3C's XML Protocols Working Group. This effectively means that SOAP is frozen and stable until such time as the W3C Working Group delivers a specification.

2.1.2. UDDI (Universal Description, Discovery and Integration Service)

UDDI provides a mechanism for clients to dynamically find other web services. Using a UDDI interface, businesses can dynamically connect to services provided by external business partners. A UDDI registry is similar to a CORBA trader, or it can be thought of as a DNS service for business applications. A UDDI registry has two kinds of clients: businesses that want to publish a service (and its usage interfaces), and clients who want to obtain services of a certain kind and bind programmatically to them. The table below is an overview of what UDDI provides [30]. UDDI is layered over SOAP and assumes that requests and responses are UDDI objects sent around as SOAP messages.

Table 2.1 UDDI

Information	Operations	Detailed information (supported by lower-level API)
White pages: Information such as the name, address, telephone number, and	Publish: How the provider of a Web service registers	Business information: Contained in a BusinessEntity object, which in turn contains information about services,

other contact information of a given business	itself.	categories, contacts, URLs, and other things necessary to interact with a given business.
<p>Yellow pages:</p> <p>Information that categorizes businesses. This is based on existing (non-electronic) standards</p>	<p>Find:</p> <p>How an application finds a particular Web service.</p>	<p>Service information:</p> <p>Describes a group of Web services. These are contained in a BusinessService object</p>
<p>Green pages:</p> <p>Technical information about the Web services provided by a given business.</p>	<p>Bind:</p> <p>How an application connects to, and interacts with, a Web service after it's been found</p>	<p>Binding information:</p> <p>The technical details necessary to invoke a Web service. This includes URLs, information about method names, argument types, and so on. The BindingTemplate object represents this data.</p> <p>Service Specification Detail:</p> <p>This is metadata about the various specifications implemented by a given Web service. These are called tModels in the UDDI specification</p>

2.1.3. WSDL (Web Services Definition Language)

WSDL provides a procedure for service providers to describe the format of web service requests. WSDL is used to describe what a web service can perform, where it is located, and how to invoke it. WSDL mostly uses SOAP/HTTP/MIME as the remote object invocation mechanism. UDDI registries describe numerous aspects of web services, including the binding details of the service. WSDL fits into the subset of a UDDI service description. UDDI registries contain WSDL documents of web services.

WSDL defines services as bundles of abstract endpoints or ports. In WSDL the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. This allows the reuse of abstract definitions of messages, which are abstract descriptions of the data being exchanged, and port types, which are abstract collections of operations. The concrete protocol and data format specifications for a particular port type constitute a reusable binding. A port is defined by associating a network address with a reusable binding; a collection of ports defines a service. And, thus, a WSDL document uses the following elements in the definition of network services:

- Types: a container for data type definitions using some type system (such as XSD).
- Message: an abstract, typed definition of the data being communicated.
- Operation: an abstract description of an action supported by the service.
- Port Type: an abstract set of operations supported by one or more endpoints.
- Binding: a concrete protocol and data format specification for a particular port type.
- Port: a single endpoint defined as a combination of a binding and a network address.
- Service: a collection of related endpoints.

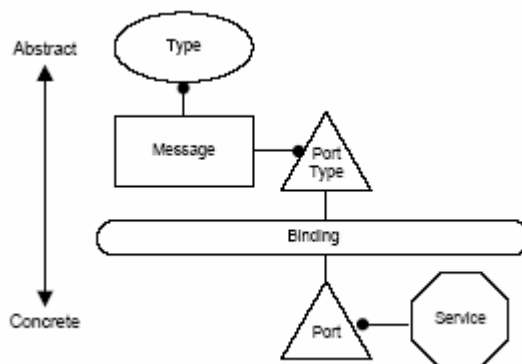


Figure 2.3 WSDL Components

2.1.3.1. Service Advertisement

A sample WSDL document, HelloWorld.wsdl:

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="HelloService"
targetNamespace="http://www.ecerami.com/wsdl/HelloService.wsdl"
```

```

xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://www.ecerami.com/wsdl/HelloService.wsdl"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <message name="SayHelloRequest">
    <part name="firstName" type="xsd:string"/>
  </message>
  <message name="SayHelloResponse">
    <part name="greeting" type="xsd:string"/>
  </message>
  <portType name="Hello_PortType">
    <operation name="sayHello">
      <input message="tns:SayHelloRequest"/>
      <output message="tns:SayHelloResponse"/>
    </operation>
  </portType>
  <binding name="Hello_Binding" type="tns:Hello_PortType">
    <soap:binding style="rpc"
      transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="sayHello">
      <soap:operation soapAction="sayHello"/>
      <input>
        <soap:body
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="urn:examples:helloservice"
          use="encoded"/>
      </input>
      <output>
        <soap:body
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="urn:examples:helloservice"
          use="encoded"/>
      </output>
    </operation>
  </binding>
  <service name="Hello_Service">
    <documentation>WSDL File for HelloService</documentation>
    <port binding="tns:Hello_Binding" name="Hello_Port">
      <soap:address
        location="http://localhost:8080/soap/servlet/rpcrouter"/>
    </port>
  </service>
</definitions>

```

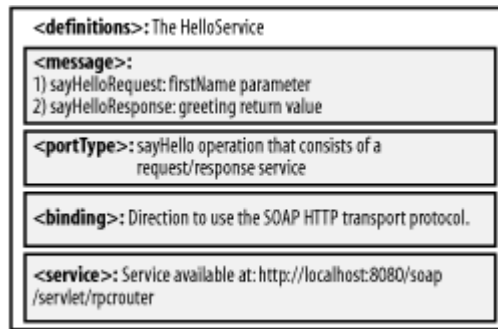


Figure 2.4 HelloWorld.wsdl

2.2. E - Business and Web Services

Internet provides extreme communication facilities. These facilities are also extremely useful for business. Electronic commerce is whole of all these business communication and also transactions over networks and through computers. Simply, electronic commerce is the buying and selling of goods and services, and the transfer of funds, through digital communications, commonly internet. Broadly electronic commerce also includes all inter-company and intra-company functions (such as marketing, finance, manufacturing, selling, and negotiation) that enable commerce and use electronic mail, EDI, file transfer, fax, video conferencing, workflow, or interaction with a remote computer.

The following items are the elements of the traditional commerce:

- Product or service to sell or buy: These products can be provided from a distributor, producer or can be produced directly.
- A place in order to realize commerce: This kind of places includes traditional stores, telephone numbers and mail addresses.
- Marketing, i.e., getting people to come to the commerce place
- A way to accept orders
- A way to accept money
- A way to deliver the product
- A way to accept returns
- Customer service

These items are exactly valid for the electronic commerce also. But in the electronic commerce, there are some advantages:

- Lower transaction costs, then lower prices: The commerce place in electronic commerce is usually a uniform resource location (URL) in the internet. So place hiring and operating costs converge to 0. Also as a result of automation, employee costs are kept in minimum.
- Larger purchases per transaction: With the relational connections amongst products, people can see many other related products in the internet. For example, during investigation of a specific book in a book sale web site, other related books can be seen easily.
- Larger catalogues: In a web site, millions of products can be hold and presented to the customers. In a traditional store, it requires a considerable area.
- Improved customer interaction: People can search and investigate the products easily on their own.

The things that are hard about e-commerce include:

- Getting traffic to come to the web site
- Getting traffic to return to your web site a second time
- Differentiating yourself from the competition
- Getting people to buy something from your web site. Having people look at your site is one thing. Getting them to actually type in their credit card numbers is another.
- Integrating an e-commerce web site with existing business data (if applicable)
- There are so many web sites, and it is so easy to create a new e-commerce web site, that getting people to look at yours is the biggest problem.
- Providing security

The things that are easy about e-commerce, especially for small businesses and individuals, include:

- Creating the web site
- Taking the orders
- Accepting payment

One of the most difficult parts of the electronic commerce is presenting the security between the buyer and the seller. Since the internet is a public media, many other external actors can be included in the transaction.

2.3. Web Service Standards

Standards development is an extremely hard and complicated process, because everything must be fairly acceptable for every interested foundation. *The best route to standards is usually for a technology provider within the appropriate space to put forward a mechanism* [31]. Inevitably, this route will be based upon the provider's own experience, knowledge and capabilities but does provide a good basis for further discussion.

A balanced approach towards standards is required if the interests of the many are to be supported rather than the commercial benefits of the few. For this reason, most standards are developed by independent bodies. For web services, there are a number of different bodies working on different aspects of the problem. The major influencers are:

- **The Worldwide Web Consortium (W3C)** - W3C is an organization committed to developing the full potential of the web. To that end, it develops interoperable technologies (specifications, guidelines, software, and tools) that will enhance the web's position as a forum for information, commerce, communication, and collective understanding. In the web services arena, W3C is working upon the wider exploitation of XML as a mechanism for exchanging information. The ultimate aim is to identify the architecture and building blocks that will enable the use of web services. (<http://www.w3c.org>)

- **Web Services Interoperability Organisation (WSI)** - perhaps the biggest contributor to the development of web services. WSI was founded early in 2002. Although the founding group was 7-strong, much of the development activity comes from IBM and Microsoft. The emphasis within WSI is to find best practices and methods that will speed up and enhance the progress of web services. WSI is specifically interested in the delivery of resources that will help to solve well-defined business problems. This comes in the form of sample implementations and code examples as well as implementation guidelines and other resources. (<http://www.ws-i.org>)

• **Internet Engineering Task Force (IETF)** - a forum for developers and other technicians to submit and develop ideas that will assist in the expansion of Internet technology. In some ways, the IETF can be seen as a resource that feeds ideas and information into the other independent bodies. (<http://www.ietf.org>)

In this part, the web services standards will be classified in two groups:

2.3.1. Core Standards

This group consists of XML, SOAP, UDDI and WSDL.

Obviously, *XML* is the dominant standard for web services. Although not only associated with web services, it is clear that XML has a major role to play in the success of any such initiatives. Its use as a universal exchange mechanism has created an ideal foundation for the development of the platform-independent web services model. In addition to the use of XML, there is general agreement on the use of three other web services standards that explained in the previous part: SOAP, UDDI and WSDL.

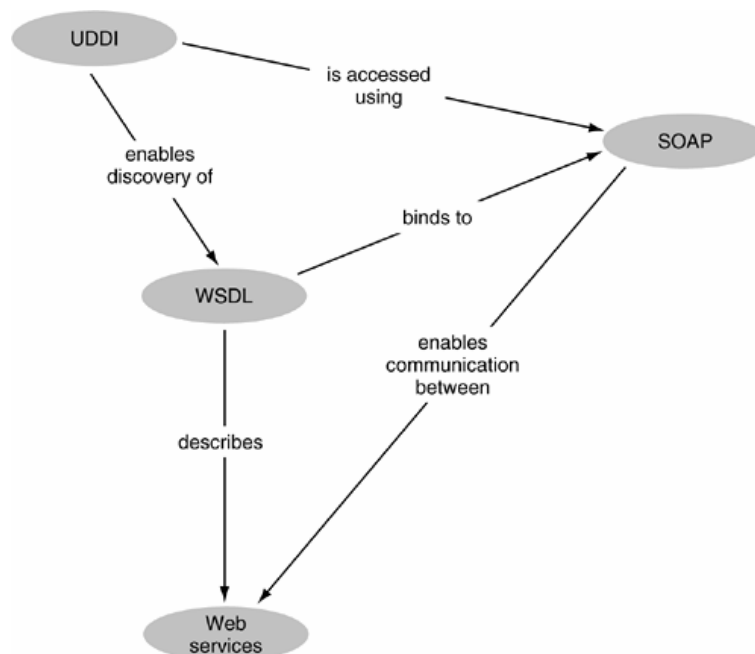


Figure 2.5 Core Web Service Standards

2.3.2. Other Standards

There are several web service standards in order to facilitate web service security and interoperability [32].

2.3.2.1. Web Services Trust Language (WS-Trust)

Version 1.1

This specification defines extensions that build on WS-Security to provide a framework for requesting and issuing security tokens, and to broker trust relationships [13].

2.3.2.2. Web Services Transaction (WS-Transaction)

This specification describes coordination types that are used with the extensible coordination framework WS-Coordination.

2.3.2.3. Web Services Coordination (WS-Coordination)

The web services coordination specification describes an extensible framework for providing protocols that coordinate the actions of the distributed applications. Such coordination protocols are used to support a number of applications, including those that need to reach consistent agreement on the outcome of distributed transactions.

2.3.2.4. Web Services Eventing (WS-Eventing)

The web services eventing specification proposes a way of communicating about events within and between Web services. The updated specification incorporates industry feedback generated from ongoing feedback and interoperability workshops and reflects progress on quality and interoperability.

2.3.2.5. Web Services Dynamic Discovery (WS-Discovery)

The web services dynamic discovery specification defines a multicast-based ad-hoc discovery protocol to locate available web services. This enables clients on a network to automatically find web services.

2.3.2.6. Web Services Federation Language (WS-Federation)

The web services federation language specification defines mechanisms that are used to enable identity, attribute, authentication, and authorization federation across different trust realms.

2.3.2.7. Web Services Reliable Messaging Protocol (WS-ReliableMessaging)

The web services reliable messaging specification describes a protocol that allows messages to be delivered reliably between distributed applications in the presence of software component, system or network failures. The protocol is described in this specification in an independent manner allowing it to be implemented using different network transport technologies. To support interoperable web services, a SOAP binding is defined in this specification.

2.3.2.8. Web Services Addressing Specification (WS-Addressing)

The web services addressing specification provides transport-neutral mechanisms to address web services and messages. Specifically this specification defines XML elements to identify web service end points and to secure end-to-end endpoint identification in messages. This specification enables messaging systems to support message transmission through networks that include processing nodes such as endpoint managers, firewalls, and gateways in a transport-neutral manner.

2.3.2.9. Web Services Policy Framework (WS-Policy)

The web services policy framework provides a general purpose model and corresponding syntax to describe and communicate the policies of a web service. WS-Policy defines a base set of constructs that can be used and extended by other web services specifications to describe a broad range of service requirements, preferences and capabilities.

2.3.2.10. Web Services Security Framework (WS-Security)

The web services security framework describes enhancements to SOAP messaging to provide quality of protection through message integrity, message confidentiality, and single message authentication.

2.3.2.11. Web Services Privacy Framework (WS-Privacy)

The web services privacy framework provides a model in order to specify how users state privacy preferences, and for how Web Services state and implement privacy practices.

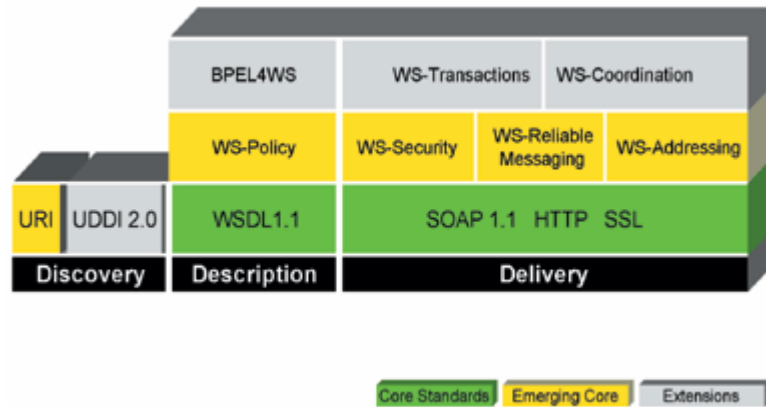


Figure 2.6 Web Services Specification Stack [32]

2.4. Privacy Protection

Privacy is the secrecy of personal information. In e-business, it is the right of individuals to determine for themselves when, how and to what extent information about them is communicated to others [4]. There must be a mechanism to balance the flow of information in the internet and the privacy of individuals. In ordinary world, people do not desire to share their personal information, because of the bad things they can face or just because they do not want. In the internet, people are more vulnerable to deceits, because they can deliver lots of information about themselves, and this delivery can be reached to unintended recipients easily.

In order to protect privacy of individuals, generally domain-specific studies have been conducted. For instance, The United States has enacted several projects to protect health care [22], finance [23] and children's data [14].

Internet users also consider their privacy. Many people are aware that giving their personally identifiable information to organizations may result in the data being used in ways the person never intended. This seems to be a great obstacle against the growth of e-business. It is believed that billions of dollars are lost in e-business because of the privacy concerns [24].

Because of that loss, some organizations are inspecting and revising their information management. They want to be seen as trustful e-business services by the potential customers. These organizations desire to be able to demonstrate that they are managing the personal data in a sensitive, trustworthy way. In addition, legislative laws encourage them to obey the

privacy rules. For example, a firm in the US paid a fine because it violated the Children's Online Privacy Protection Act [14]. In order to keep their customers, organizations allow their users to specify their privacy policies and obey those documents.

2.4.1. A Privacy Model For Web Services

E-business privacy can be investigated in three parts: user privacy, service privacy, and data privacy [5]. In the thesis, user privacy and service privacy issues shall be taken into consideration.

2.4.1.1. User Privacy

Individuals and applications that work on behalf of them are the users of the web services. Users may need different privacy levels for different information domains. For example, credit card numbers may be desired to be more restricted than address information. The other factors that affect the privacy policies are the recipient list and the retention purpose of the personal information. These privacy preferences build up privacy profiles of the users. These profiles can be created, updated, and deleted by the privileged users, for general cases, by themselves. Individuals or groups of individuals may be assigned privacy profiles.

2.4.1.2. Service Privacy

Generally, web services define generic privacy rules for all users. These rules constitute three types of policy: usage policy, storage policy, and disclosure policy. The usage policy states the purposes for which the information collected can be used. The storage policy defines where and until when the data is stored. The disclosure policy defines to whom the data shall be delivered further. For example, the usage policy may state that the data is stored for administrative purposes, the storage policy may state that the data shall be stored for a month, and finally the disclosure policy may state that the data shall be delivered to third party sites.

2.4.1.3. Data Privacy

Data privacy is out of the scope of this thesis work. It is established on the fact that more than one web services may use the same data item on the same database. This type of policy tries to regulate the access to the various data items in databases.

2.4.2. Privacy Protection Tools

Online privacy protection tools can be grouped as seen in the following picture [7]:

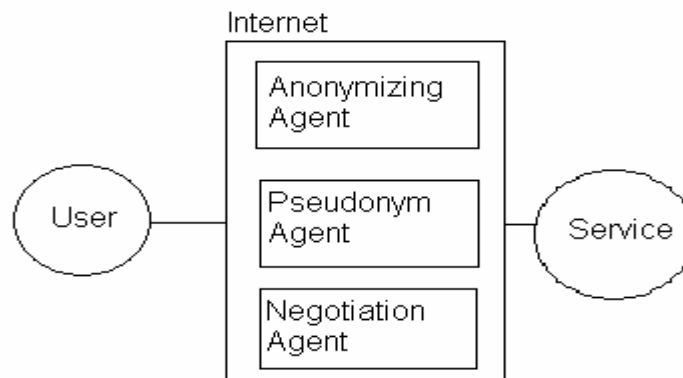


Figure 2.7 Privacy Protection Tools

Anonymizing agents are services that submit requests to web sites on behalf of users [15]. These agents work as the proxies. IP addresses of users become invisible, web services see the IP address of the agents.

Pseudonym agents provide nicknames or encrypted identifiers for users so that they cannot be identified by the service providers.

Negotiation agents check whether privacy policies of the service are appropriate on behalf of the user.

In the thesis, the main concern is the design and implementation of a P3P negotiation agent.

2.5. Ontology Concept

Ontology is the most fundamental subject of metaphysics in philosophy. It studies existence and tries to find out entities that exist. In computer science, an ontology is used to express a conceptual schema of a given domain. It includes entities and their relations in the domain.

With the relations and rules expressed in the ontology, data can gain useful meaning. Meaningful data is the basis of semantic programming, such as semantic web. Semantic web then converges to more interoperable framework, in which computers can search and gain

data automatically in the internet without the need of users. Considering this goal, internet is supposed to be oriented towards the semantic web [3].

In the thesis, ontology documents (OWL documents [19]) are thought to be the data sources of privacy documents. Message ontologies are parsed with the Jena parser [21], and then data items are extracted from the parsed ontology documents. In the work, privacy attributes are assigned to these data items.

2.5.1. OWL

OWL is the abbreviation of Web Ontology Language. It is built on the RDF (Resource Description Framework [43]) syntax, but it is stronger than RDF. It is stronger because, it has a larger vocabulary and a stronger syntax.

OWL is presented in three types, which are more expressive subsequently for different purposes:

2.5.1.1. OWL Lite

OWL Lite is designed and prepared for the beginner users that need a classification hierarchy and simple constraints [20]. For example, it only permits cardinality values of 0 or 1. It is supposed to be simpler to provide tool support for OWL Lite than its more expressive relatives.

2.5.1.2. OWL DL

OWL DL provides complete and decidable computations with the maximum expressiveness. In OWL DL, users can use all of the OWL constructs under some restrictions. For instance, while a class may be a subclass of more than one classes, a class cannot be an instance of another class. DL in OWL DL means description logics, which is the formal base of OWL.

2.5.1.3. OWL Full

OWL Full is the most expressive version of OWL that provides the syntactic freedom of RDF with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right. Reasoning software products does not seem to be able to support complete reasoning for every feature of OWL Full.

2.5.2. JENA Library

Jena [21] is a Semantic Web framework developed in Java programming language. It handles RDF and OWL documents and provides a rule-based inference engine.

Jena is an open source project supported by the HP Labs Semantic Web Programme.

The Jena Framework includes:

- A RDF API
- Reading and writing RDF in RDF/XML, N3 and N-Triples
- An OWL API
- In-memory and persistent storage
- RDQL – a query language for RDF

2.6. Platform for Privacy Preferences Project (P3P)

Platform for Privacy Preferences Project (P3P) enables web service providers and web service requestors to declare their privacy preferences. Those privacy preferences are in a standard format so that they can be easily processed. Thus, service requestors do not need to read the privacy policies at every service they use. Also these preferences must be easily interpreted by the users.

The P3P specification includes a standard vocabulary for describing a web site's data practices, a set of base data elements that web sites can refer to in their P3P privacy policies, and a protocol for requesting and transmitting web site privacy policies [16]. P3P policies are encoded in a machine-readable XML format using the P3P vocabulary. The P3P protocol is a simple extension to the HTTP protocol used for fetching web pages. P3P user agents use standard HTTP requests to fetch a P3P policy reference file from a well-known location on the web site to which a user is making a request. The policy reference file indicates the location of the P3P policy file that applies to each part of the web site. There might be one policy for the entire site, or several different policies that each covers a different part of the site. The user agent can then fetch the appropriate policy, parse it, and take action according to the user's preferences.

The first major commercial user agent implementation of P3P appeared in the Microsoft Internet Explorer web browser in the summer of 2001 [12]. In January 2002 the P3P Specification working group released a stable “Proposed Recommendation” draft of the P3P specification. This draft was the result of over five years of work and took into account the suggestions of many individuals who had commented on earlier public drafts. A final P3P 1.0 “Recommendation” was emerged in April 2002 after the W3C members review and vote on the Proposed Recommendation [6].

Although P3P provides a technical mechanism for ensuring that users can be informed about privacy policies before they release personal information, it does not provide a technical mechanism for making sure that sites act according to their policies. In addition, P3P does not include mechanisms for transferring data or for securing personal data in transit or storage.

P3P Elements

Adapted from [2]:

- POLICIES: Combines one or more policies together in a single file. A ‘POLICIES’ element is the root element of policy files.
- POLICY: Identifies and encapsulates a complete WS-Privacy.
- TEST: Is used to show that WS-Privacy document is a test document.
- ACCESS: Indicates the web service allows access to the specified types of information. This field must have one of the following values:
 - non-ident: Web service does not contain identified data
 - all: Access is allowed to all identified data
 - contact-and-other: Access is given to online and physical contact information and some other identified data
 - ident-contact: Access is given to online and physical contact information
 - other-ident: Access is given to some other identified data such as users’ account charges
 - none: No access is allowed to identified data

- DISPUTES: Defines dispute resolution methods. Each privacy document should have at least one DISPUTES element. Disputes can occur when services do not obey their privacy preferences and violates protocols. This element has attributes resolution-type, service, short-description and elements LONG-DESCRIPTION and REMEDIES.
 - resolution-type: Has one of the following values:
 - Customer Service (service): Users can complain about information misuse and distribution to the web service customer service representative
 - Independent Organization (independent): Users can complain to independent third party organizations
 - Court (court): Users can file a legal complaint against the web service
 - Applicable Law (law): Disputes will be resolved in accordance with the law specified
 - service: URI of the customer service web page, independent organization, information about the relevant court or information about the applicable law
 - verification: URI or certificate that can be used for verification purposes
 - short-description: A short human readable description with no more than 255 characters
 - LONG-DESCRIPTION: Encapsulates a human readable long description about dispute-resolving.
- REMEDIES: Defines what to do in case of a privacy violation. Each DISPUTES element should contain a REMEDIES element. The REMEDIES element must have one or more of the following remedies:
 - correct: Errors will be corrected by the service

- money: A specified amount in the human readable description or amount of damage will be paid to the user
 - law: Remedies will be determined by the law stated in the human readable description
- STATEMENT: Contains data to which the privacy preferences will be assigned. It combines a PURPOSE element, a RECIPIENT element, a RETENTION element, a DATA-GROUP element and optionally a CONSEQUENCE element and a NON-IDENTIFIABLE element.
- CONSEQUENCE: Consequences that can be shown to a human user to explain why the suggested practice may be valuable in a particular instance even if the user would not normally allow the practice.
- NON-IDENTIFIABLE: This element signifies that either no data is collected (including Web logs), or that the organization collecting the data will anonymize the data referenced in the enclosing STATEMENT. In order to consider the data anonymized, there must be no reasonable way for the entity or a third party to attach the collected data to the identity of a natural person. Some types of data are inherently anonymous, such as randomly-generated session IDs. Data which might identify natural people in some circumstances, such as IP addresses, names, or addresses, must have a non-reversible transformation applied in order to be considered anonymized. An example of a non-reversible transformation is removing the last seven bits of an IP address and replacing them with zeros. This transformation must be applied to all copies of the data, including those that might be stored on backup media. An algorithm that replaces identified data with unique corresponding values from a table is not considered non-reversible. In addition, a one-way cryptographic hash would not be considered non-reversible if the set of possible data values is small enough that all possible hashed values can be generated and compared with the value that someone is attempting to reverse.
- PURPOSE: Each statement must contain a PURPOSE element that includes one or more purposes of data collection or uses of data. The purpose element must contain one or more of the following purposes. Each type of purpose (with the exception of current) can have the optional attribute 'required':

- current: Completion and Support of Activity For Which Data Was Provided: Information may be used by the service provider to complete the activity for which it was provided, whether a one-time activity such as returning the results from a Web search, forwarding an email message, or placing an order; or a recurring activity such as providing a subscription service, or allowing access to an online address book or electronic wallet.
- admin: Information may be used for the technical support of the Web site and its computer system. This would include processing computer account information, information used in the course of securing and maintaining the site, and verification of Web site activity by the site or its agents.
- develop: Information may be used to enhance, evaluate, or otherwise review the site, service, product, or market. This does not include personal information used to tailor or modify the content to the specific individual nor information used to evaluate, target, profile or contact the individual.
- tailoring: Information may be used to tailor or modify content or design of the site where the information is used only for a single visit to the site and not used for any kind of future customization. For example, an online store might suggest other items a visitor may wish to purchase based on the items he has already placed in his shopping basket.
- pseudo-analysis: Information may be used to create or build a record of a particular individual or computer that is tied to a pseudonymous identifier, without tying identified data (such as name, address, phone number, or email address) to the record. This profile will be used to determine the habits, interests, or other characteristics of individuals for purpose of research, analysis and reporting, but it will not be used to attempt to identify specific individuals. For example, a marketer may wish to understand the interests of visitors to different portions of a Web site.

- pseudo-decision: Information may be used to create or build a record of a particular individual or computer that is tied to a pseudonymous identifier, without tying identified data (such as name, address, phone number, or email address) to the record. This profile will be used to determine the habits, interests, or other characteristics of individuals to make a decision that directly affects that individual, but it will not be used to attempt to identify specific individuals. For example, a marketer may tailor or modify content displayed to the browser based on pages viewed during previous visits.
- individual-analysis: Information may be used to determine the habits, interests, or other characteristics of individuals and combine it with identified data for the purpose of research, analysis and reporting. For example, an online Web site for a physical store may wish to analyze how online shoppers make offline purchases.
- individual-decision: Information may be used to determine the habits, interests, or other characteristics of individuals and combine it with identified data to make a decision that directly affects that individual. For example, an online store suggests items a visitor may wish to purchase based on items he has purchased during previous visits to the Web site.
- contact: Information may be used to contact the individual, through a communications channel other than voice telephone, for the promotion of a product or service. This includes notifying visitors about updates to the Web site. This does not include a direct reply to a question or comment or customer service for a single transaction -- in those cases, <current/> would be used. In addition, this does not include marketing via customized Web content or banner advertisements embedded in sites the user is visiting -- these cases would be covered by the <tailoring/>, <pseudo-analysis/> and <pseudo-decision/>, or <individual-analysis/> and <individual-decision/> purposes.
- historical: Information may be archived or stored for the purpose of preserving social history as governed by an existing law or policy. This law or policy must be referenced in the <DISPUTES> element

and must include a specific definition of the type of qualified researcher who can access the information, where this information will be stored and specifically how this collection advances the preservation of history.

- telemarketing: Information may be used to contact the individual via a voice telephone call for promotion of a product or service. This does not include a direct reply to a question or comment or customer service for a single transaction -- in those cases, <current/> would be used.
- other-purpose: Information may be used in other ways not captured by the above definitions. (A human readable explanation must be provided in these instances).
- required: Whether the purpose is a required practice for the site. The attribute can take the following values:
 - always : The purpose is always required; users cannot opt-in or opt-out of this use of their data. This is the default when no required attribute is present.
 - opt-in : Data may be used for this purpose only when the user affirmatively requests this use -- for example, when a user asks to be added to a mailing list. An affirmative request requires users to take some action specifically to make the request. For example, when users fill out a survey, checking an additional box to request to be added to a mailing list would be considered an affirmative request. However, submitting a survey form that contains a pre-checked mailing list request box would not be considered an affirmative request. In addition, for any purpose that users may affirmatively request, there must also be a way for them to change their minds later and decline.
 - opt-out : Data may be used for this purpose unless the user requests that it not be used in this way. When this value is selected, the service must provide clear instructions to users on how to opt-out of this purpose. Services should also provide these instructions or a pointer to these instructions at the point of data collection.

- RECIPIENT: STATEMENT element must contain one or more RECIPIENT elements that references one or more recipients of the selected data. The RECIPIENT elements must have one or more of the following recipients:
 - ours: Service providers and/or entities acting as their agents or entities for whom they are acting as an agent.
 - delivery: Legal entities performing delivery services that may use data for purposes other than completion of the stated purpose. This should also be used for delivery services whose data practices are unknown.
 - same: Legal entities who use the data on their own behalf under equitable practices.
 - other-recipient: Legal entities following different practices: Legal entities that are constrained by and accountable to the original service provider, but may use the data in a way not specified in the service provider's practices
 - unrelated: Legal entities whose data usage practices are not known by the original service provider.
 - public: Public areas such as bulletin boards, public directories, or commercial CD-ROM directories.

- RETENTION: Each STATEMENT element must contain a RETENTION element that indicates the kind of retention policy that applies to the data referenced in that statement. The RETENTION element must contain one of the following retention values:
 - no-retention: Information is not retained for more than a brief period of time necessary to make use of it during the course of a single online interaction. Information must be destroyed following this interaction and must not be logged, archived, or otherwise stored. This type of retention policy would apply, for example, to services that keep no Web server logs, set cookies only for use during a single session, or collect information to perform a search but do not keep logs of searches performed.

- stated-purpose: Information is retained to meet the stated purpose. This requires information to be discarded at the earliest time possible. Sites must have a retention policy that establishes a destruction time table. The retention policy must be included in or linked from the site's human-readable privacy policy.
- legal-requirement: Information is retained to meet a stated purpose, but the retention period is longer because of a legal requirement or liability. For example, a law may allow consumers to dispute transactions for a certain time period; therefore a business may for liability reasons decide to maintain records of transactions, or a law may affirmatively require a certain business to maintain records for auditing or other soundness purposes. Sites must have a retention policy that establishes a destruction time table. The retention policy must be included in or linked from the site's human-readable privacy policy.
- business-practices: Information is retained under a service provider's stated business practices. Sites must have a retention policy that establishes a destruction time table. The retention policy must be included in or linked from the site's human-readable privacy policy.
- indefinitely: Information is retained for an indeterminate period of time. The absence of a retention policy would be reflected under this option. Where the recipient is a public area, this is the appropriate retention policy.
- DATA-GROUP: Each STATEMENT element must include one or more DATA-GROUP element. Each DATA-GROUP element must contain one or more DATA elements.
- DATA: Describes the data to be transferred. It has a mandatory attribute 'ref'.
 - ref: URI reference of the data.

The following is an example privacy policy document:

```
<POLICIES xmlns="http://www.w3.org/2002/01/P3Pv1">
```



```

<POLICY name="forShoppers"
discuri="http://www.catalog.example.com/Privacy/PrivacyPracticeShopping.html"
opturi="http://catalog.example.com/preferences.html"
  xml:lang="en">
<ENTITY>
  <DATA-GROUP>
    <DATA ref="#business.name">CatalogExample</DATA>
    <DATA ref="#business.contact-info.postal.street">4000 Lincoln Ave.</DATA>
  </DATA-GROUP>
</ENTITY>
<ACCESS><contact-and-other/></ACCESS>
<DISPUTES-GROUP>
  <DISPUTES resolution-type="independent"
  service="http://www.PrivacySeal.example.org"
  short-description="PrivacySeal.example.org">
  <IMG src="http://www.PrivacySeal.example.org/Logo.gif" alt="PrivacySeal's
logo"/>
  <REMEDIES><correct/></REMEDIES>
</DISPUTES>
</DISPUTES-GROUP>
<STATEMENT>
  <CONSEQUENCE>
    We record some information in order to serve your request
    and to secure and improve our Web site.
  </CONSEQUENCE>
  <PURPOSE><admin/><develop/></PURPOSE>
  <RECIPIENT><ours/></RECIPIENT>
  <RETENTION><stated-purpose/></RETENTION>
  <DATA-GROUP>
    <DATA ref="#dynamic.clickstream"/>
    <DATA ref="#dynamic.http.useragent"/>
  </DATA-GROUP>
</STATEMENT>
<STATEMENT>
  <CONSEQUENCE>
    We use this information when you make a purchase.
  </CONSEQUENCE>

```

```
<PURPOSE><current/></PURPOSE>
<RECIPIENT><ours/></RECIPIENT>
<RETENTION><stated-purpose/></RETENTION>
<DATA-GROUP>
  <DATA ref="#user.name"/>
  <DATA ref="#user.home-info.postal"/>
  <DATA ref="#user.login.id"/>
  <DATA ref="#user.login.password"/>
  <DATA ref="#dynamic.miscdata">
  </DATA>
</DATA-GROUP>
</STATEMENT>
</POLICY>
</POLICIES>
```

2.7. Policy Management

Since privacy documents are supposed to be integrated into policy documents, web service policy (WS - Policy) needs more clarification than other web service standards.

WS-Policy provides a flexible and extensible grammar and a vocabulary for expressing the capabilities, requirements, and general characteristics of entities in an XML Web Services-based system [27]. WS-Policy defines a framework and a model for the expression of these properties as policies. Policy expressions allow for both simple declarative assertions and more sophisticated conditional assertions.

In WS-Policy, policy is a collection of policy assertions. Some of these assertions state requirements and capabilities to select the proper service. Some of them specify requirements and capabilities about more low-level issues (e.g., authentication scheme, transport protocol selection). WS-Policy provides a single policy grammar for both kinds of assertions. Unfortunately, WS-Policy is insufficient to specify how policies are discovered or attached to a Web service.

```
<wsp:Policy>
  <wsp:ExactlyOne>
    <wsse:SecurityToken>
      <wsse:TokenType>wsse:Kerberosv5TGT</wsse:TokenType>
    </wsse:SecurityToken>
    <wsse:SecurityToken>
      <wsse:TokenType>wsse:X509v3</wsse:TokenType>
    </wsse:SecurityToken>
  </wsp:ExactlyOne>
</wsp:Policy>
```

Figure 2.8 Sample Policy

2.7.1. Policy Assertions

A policy assertion contains a requirement, a capability, or another property of a behavior. Assertions indicate domain-specific (e.g., security, transactions) semantics and are expected to be defined in separate, domain-specific specifications.

Policy assertions can have 'Usage' attribute. Potential values of this attribute are listed in the table below:

Table 2.2 Usage Values

Value	Meaning
Required	The assertion must be applied to the subject. If the subject does not meet the criteria expressed in the assertion, a fault or error will occur.
Rejected	The assertion is explicitly not supported and if present will cause failure.
Optional	The assertion may be made of the subject but it is not required to be applied.
Observed	The assertion shall be applied to all subjects and requesters of the service are informed that the policy will be applied.
Ignored	The assertion is processed, but ignored. That is, it can be specified, but no action will be taken because of it being specified. Subjects and requesters are informed that the policy will be ignored.

2.7.2. Policy Expressions

A policy expression is a collection of policy assertions that are connected by policy operators. A policy expression can represent a policy document.

2.7.3. Policy Operators

Policy operators are used to specify alternative acceptable policy assertions. The operators are:

- <wsp:All> which requires that all of its child elements be satisfied
- <wsp:ExactlyOne> which requires that exactly one of its child elements be satisfied
- <wsp:OneOrMore> which requires that at least one of its child elements be satisfied
- <wsp:Policy> whose semantics are the same as <wsp:All>

2.7.4. Other Policy Implementations

Web Service Policy Language [26] is designed to express a wide range of policies such as authorization, reliable messaging, and quality-of-service. It can merge two policies, and define policies on comparison processes. In WSPL, policies are composed of rules listed with respect to their precedence values. Rules are defined as sequences of predicates such as equals, greater than or less than.

Here is an example policy defined in WSPL:

```
Policy (Aspect = "Quality of Protection") {  
  
  Rule {  
  
    Signature-Algorithm = "RSA-SHA1",  
  
    Key-Length >= 2048 }  
  
  Rule {  
  
    Signature-Algorithm = "RSA-SHA1",  
  
    Key-Length >= 1024,  
  
    Source-Domain = "foo.com" }  
  
  Rule {  
  
    Source-Domain = "foo.com" }  
  
}
```

Work in [18] tries to provide a method to define a new policy and access control framework for business processes. In [25], an access control methodology and modular security policies for web-based applications are proposed.

2.8. SATINE Project

The implementation, which is the product of this thesis, will be deployed in a European Commission project, SATINE [33]. SATINE is the abbreviation of ‘Semantic-based Interoperability Infrastructure for Integrating Web Service Platforms to Peer-to-Peer Networks’.

SATINE project is managed by the METU Software Research and Development Center (SRDC). There are several sub-contractors from different European countries in the project.

SATINE project aims to make tourism information services cheaper and more interoperable with semantically enriched web services. Today, the travel information services are dominantly provided by Global Distribution Systems (GDS), which provide access to real time availability and price information for flights, hotels and car rental companies [34]. However, GDSs have legacy architectures with private networks, specialized hardware, and limited speed and search capabilities. Another disadvantage about GDSs is that small and medium sized companies cannot use them because of their expensive services. Furthermore, it is very difficult to interoperate them with other systems and data sources. For these reasons, web service technology, which offers interoperability on ubiquitous media and internet, is an ideal fit for travel information systems.

Several companies have applied web services on the travel information systems, but the semantics of the services is not attempted. However to be able to exploit Web services to their full potential, it is necessary to introduce semantics. Without describing the semantics of web services, it is difficult to find them in an automated way.

The following list summarizes the utilities of the SATINE project:

- SATINE project has its own P2P communication framework over the Internet
- SATINE project enables users to prepare their own web services
- SATINE project enables service discovery
- SATINE project enables user management
- SATINE project enables complex service composition via the semantics of the services.

- SATINE project wraps already existing applications as web services

The SATINE Project has realized a secure semantic-based interoperability framework for exploiting Web service platforms in conjunction with Peer-to-Peer networks in tourism industry. SATINE uses an open industry standard specification, OTA [39], compliant messages between the web service peers. In the thesis work, message ontologies developed by OTA have been used.

2.8.1. The Open Travel Alliance (OTA)

In order to manage and develop e-business on the travel domain, the travel industry has formed a consortium called the Open Travel Alliance (OTA). OTA is producing XML schemas of message specifications to be exchanged between the trading partners, including availability checking, booking, rental and reservation.

The travel industry needs common technical specifications for the electronic flow of information in order to use ubiquitous communication channels such as the internet. Companies from all sectors of the tourism domain have come together to build these domain-wide common specifications and technologies. These technologies enable travel companies to exchange data to construct interoperable systems. Interoperable systems mean more complex travel web services that response better to demands of users. OTA tries to realize this interoperability through standard message definitions described in a common language, XML. These messages are built by the technical and industrial organizations.

The OTA specification can be thought as a common travel language that aims to:

- Develop the communication protocols for the travel domain
- Determine messages that will be exchanged amongst tourism partners and customers
- Extend if needed

Examples of OTA messages are listed below. The first one represents a flight status request; the second one is the response for the request.

Request:

```
<OTA_AirFlifoRQ xmlns="http://www.opentravel.org/OTA/2003/05"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opentravel.org/OTA/2003/05
OTA_AirFlifoRQ.xsd" EchoToken="566732" TimeStamp="2003-03-17T09:30:47-
05:00" Target="Production"
Version="1.000" SequenceNمبر="1">
```

```
<Airline Code="DL"/>
<FlightNumber>1716</FlightNumber>
<DepartureDate>2003-04-15</DepartureDate>
<DepartureAirport LocationCode="ATL"/>
<ArrivalAirport LocationCode="MSP"/>
</OTA_AirFlifoRQ>
```

Response:

```
<OTA_AirFlifoRS EchoToken="566732" TimeStamp ="2003-03-17T09:30:47-
05:00" Target=" Production"
Version="1.000" SequenceNbr="1" xmlns
="http://www.opentravel.org/OTA/2003/05"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opentravel.org/OTA/2003/05
OTA_AirFlifoRS.xsd">
<Success/>
<FlightInfoDetails TotalFlightTime="PT2H38M" TotalMiles="907">
<FlightLegInfo FlightNumber="1716" JourneyDuration="PT2H38M"
LegDistance="907">
<DepartureAirport LocationCode="ATL"/>
<ArrivalAirport LocationCode="MSP" Diversion="0"/>
<MarketingAirline CompanyShortName=" Delta" Code="DL">Delta Airlines
</MarketingAirline>
<Equipment AirEquipType="738">Boeing 737-800</Equipment>
<DepartureDateTime Scheduled="2003-04-15T08:05:00"/>
<ArrivalDateTime Scheduled="2003-04-15T09:43:00" Estimated="2003-04-
15T09:43:00"/>
</FlightLegInfo>
</FlightInfoDetails >
</OTA_AirFlifoRS >
```


CHAPTER 3

3. DESIGN OF THE WEB SERVICE PRIVACY FRAMEWORK

In the design phase, Unified Modeling Language [40] (UML) diagrams were drawn. The design was directed according to the future use of the implementation by a project supported by the European Commission. A graphical user interface is provided in order to prepare web service privacy documents easily. Also it should enable users to open and modify existing privacy documents. In order to keep the interoperability with the services all over the world, World Wide Web Consortium P3P specification is selected as the base privacy content.

3.1. Use Cases

A service provider is the user who manages a web service and its privacy requirements. If a web service needs personal information about the service requestor, in order to inform him/her, the provider should prepare a privacy document and provide it.

A service requestor is the user who wants to use a web service. He/she can declare his/her own privacy choices in order to see if the service requires an information which he/she does not deliver.

The service provider first prepares the service's privacy requirements and choices. For instance, 'Payment' service may require telephone number of the customer. He/she deploys the privacy document in a 'specified' location. Meanwhile the service requestor prepares his/her own privacy document. For example he/she may not desire to make his/her credit card number public.

When the requestor initiates the service call, first their privacy documents are compared by the server. If the documents do not match, then the process is terminated, else the process is continued.

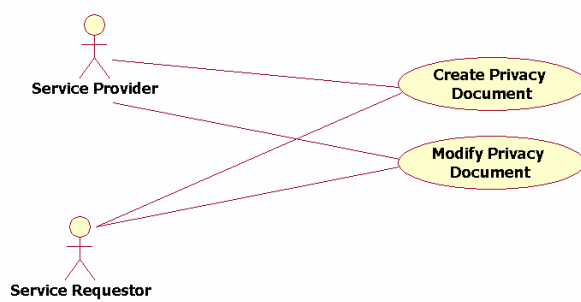


Figure 3.1 Create and modify use case diagrams

As seen in the Figure 3-1, both the service provider and the service requestor must be able to prepare a new privacy document, and modify an existing one. Creating the document must contain selecting the base ontology, providing the privacy policy content, and saving it in a proper format.



Figure 3.2 Matching use case diagram

Matching process must be realized by the web service, or the server.

3.2. Sequence Diagrams

3.2.1. Service Provider Creates a Privacy Document Sequence

After starting the program, the service provider chooses to create a new privacy document. He/she selects the service provider mode, and then starts to prepare the document. He/she selects the base ontology upon which the document will be constructed. Then the provider starts to input the privacy information. First he/she begins preparing the statement base, i.e. statement templates. During this process, purpose, retention, recipient and consequence fields of the templates are filled in by the user. Then, those templates are assigned to the ontology nodes. The desired dispute information is provided and if required rules are defined as the final step. Then the privacy policy is saved as an XML document.

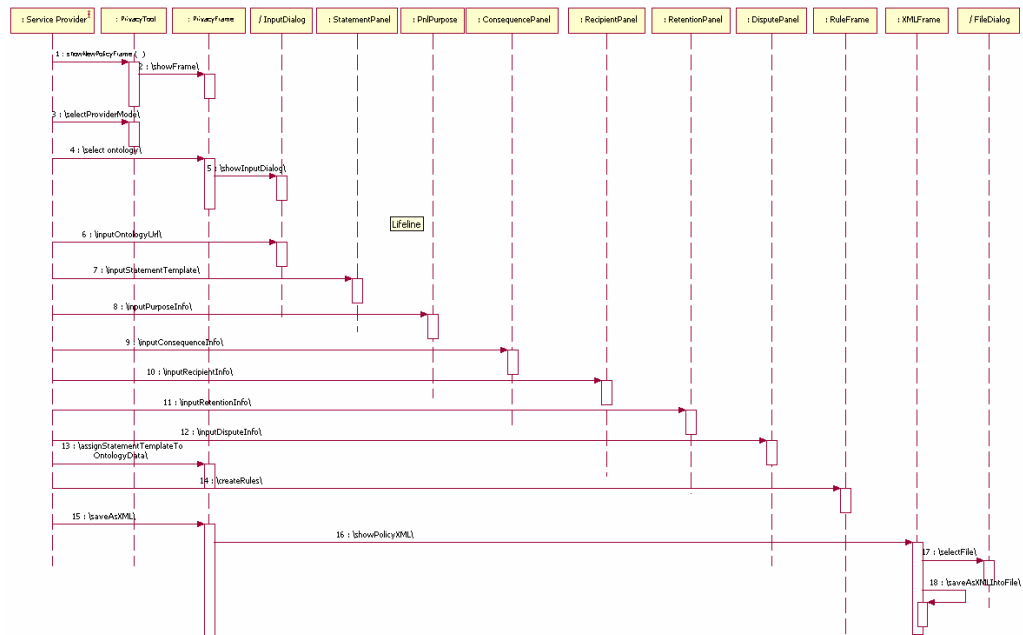


Figure 3.3 Service provider creates a privacy document sequence diagram

3.2.2. Service Requestor Creates a Privacy Document Sequence

After initiating the program, the service requestor chooses to create a new privacy document. He/she selects the service requestor mode, and then starts to prepare the document. He/she selects the base ontology upon which the document will be constructed. Then the provider starts to input the privacy information. First he/she begins preparing the statement base, i.e. statement templates. During this process, purpose, retention, recipient and consequence fields of the templates are filled in by the user. Then, those templates are assigned to the ontology nodes. The desired dispute information is provided and if required rules are defined as the final step. Then the privacy policy is saved as an XML document.

Process is almost same with the service provider case. As one of the differences, program runs as the requestor mode. The user should select the service requestor mode. The other difference is about the rules. In requestor mode, rules cannot be created.

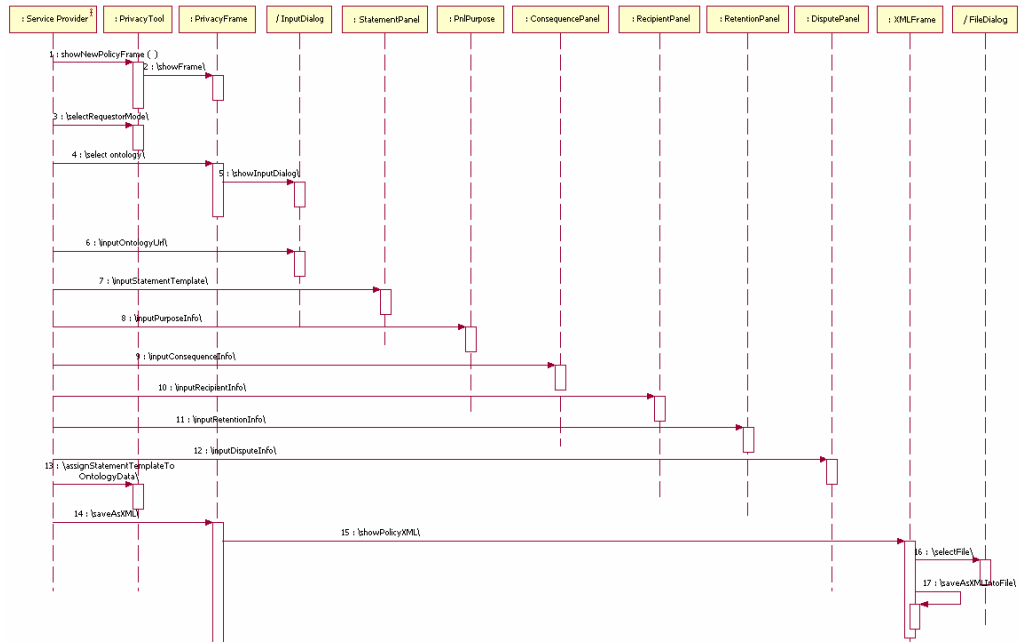


Figure 3.4 Service requestor creates a privacy document sequence diagram

3.2.3. Service Provider Modifies an Existing Privacy Document

After starting the program, the service provider chooses to open an existing privacy document. He/she selects the existing document via the file dialog, and then starts modifying the document. He/she modifies the document as in the creation phase. When the modification is completed, the user saves the new document to an XML file.

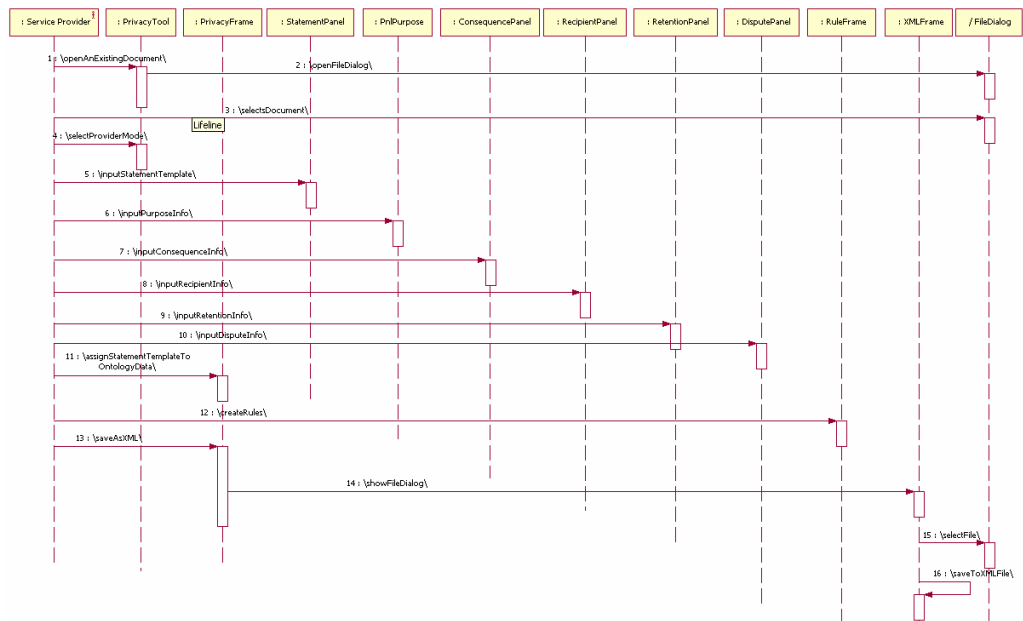


Figure 3.5 Service provider modifies an existing privacy document sequence diagram

3.2.4. Service Requestor Modifies an Existing Privacy Document

The same procedure with the modification of the service provider is applied. Only differences are the program mode and the non-existence of the ability to define rules.

3.2.5. Compare Privacy Documents

In this scenario, the user starts the comparison module of the program. He/she selects the provider and the requestor privacy documents. After selection, he/she wants to see if these privacy documents match. As a result, the user sees a dialog that shows the result of the comparison.

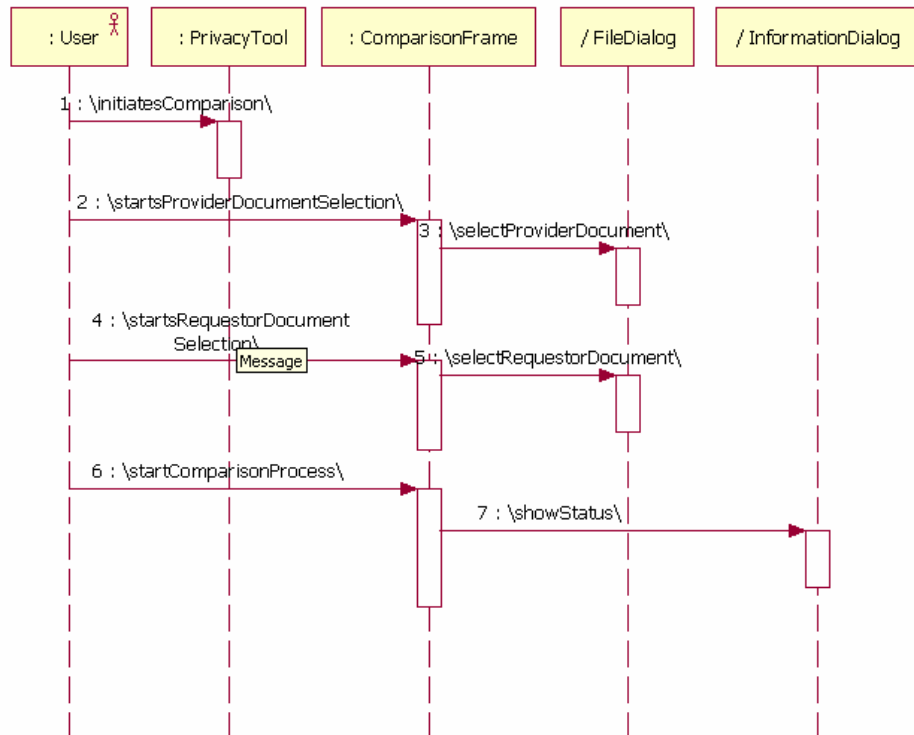


Figure 3.6 Privacy matching sequence diagram

3.3. Class Diagrams

Class diagrams are presented in Appendix – A.

CHAPTER 4

4. IMPLEMENTATION OF THE WEB SERVICE PRIVACY FRAMEWORK

4.1. Work Done

In the design phase, Unified Modeling Language [40] (UML) diagrams were drawn. The design was implemented according to the future use of the implementation by a project supported by the European Commission. Rational XDE Developer for Java [41] was used as the design platform. Use case, sequence, and class diagrams were drawn in UML format. Class diagrams can be reached at Appendix A.

A graphical user interface should be provided in order to prepare web service privacy documents easily. Also it should enable users to open and modify existing privacy documents. In order to keep the interoperability with the services all over the world, World Wide Web Consortium privacy framework specification [2] was selected the base privacy content.

The thesis was implemented in Java programming language [42]. Since the Java provides platform-independency, it complies with the purpose of the web services. In any operating system that web services can be deployed, the web service privacy framework should work properly.

As the graphical user interface, Java Swing package of the standard Java software development kit was used. It was able to fulfill all the graphical interface requirements of the thesis.

As the ontology parser, Jena 2 semantic web framework has been used [10]. Using the base classes of the Jena 2 ontology sub system, a parser has been written in Java programming language.

4.2. Detailed Explanation

4.2.1. Usage

This work aims to enable service providers and service requestors to prepare privacy documents. These documents will be used just before the service call starts. Before the initiation of the web service, privacy document of the client side is extracted from the SOAP message header. Then this document is compared with the server's privacy document on the server side. If the privacy documents match, the service call is continued, else the server side truncates the negotiation.

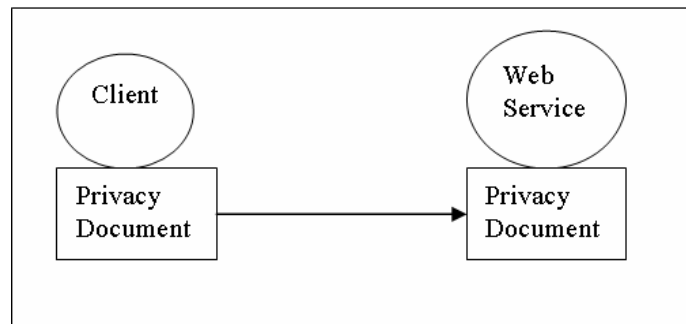


Figure 4.1 Privacy Document Usage

Prepared privacy document is thought to be embedded into the 'Policy' part in the SOAP message header. So with the first message transfer, the system can understand whether the server's and the client's privacies are appropriate. With the response message, the result can be delivered to the client side.

Web services can be used with their full potential if they can be processed with their semantics. Dealing with the semantics of web services is useful in several respects such as giving meaning to the services or enabling automated service discovery and composition. Privacy protection is another concern that can benefit from the semantics of Web services. There are some important considerations in developing privacy mechanisms:

- Only the minimal set of 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.

4.2.2. Privacy Policy Management

The data defined by a web service is composed of four 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 privacy attributes defined with respect to privacy specification of W3C. The third one is the declaration of policy attributes, i.e. how essential the data is for the service to execute. Those values are ‘Mandatory’ and ‘Optional’. 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 must provide her postal address. Alternatives may help to reach an agreement during negotiation [1].

The privacy tool can work in two separate modes. The first one is the service provider mode. In that mode, users can define privacy preferences from a service provider’s point of view. They are allowed to state the web service’s privacy requirements.

The second mode is the service requestor mode. Using the program in that mode, users can declare the privacy preferences of the client services. They can state the privacy preferences from a service requestor’s point of view. They define their privacy preferences for Web services through a rule-based mechanism with a reference to 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.

During negotiation phase, those criteria are used:

Table 4.1 Data Policy Attributes and System Behaviors

Server Privacy Request	Client Privacy Delivery	Do Privacy Documents Match?	Is Data Delivered?
Mandatory	Free	Yes	Yes
Mandatory	Limited	Yes	Yes
Mandatory	Not Given	No	No
Optional	Free	Yes	Yes
Optional	Limited	Yes	No
Optional	Not Given	Yes	No

Two privacy documents do not match only if one data item is necessary for the service and it is not given by the user. In other circumstances, those documents can match.

Data is not delivered in case it is not given by the user; or its delivery is limited and it is an optional data item for the service. In other cases, data item is delivered by the client software.

4.2.3. Preparing Documents

Since privacy policies are based on domain ontologies, first a domain ontology must be selected in order to build a document. Then the selected ontology (OWL document) is parsed to compose a tree structure.

First an OWL document is chosen by a file dialog. Then selected document is parsed by the Jena library [21], producing an ontology model. Then this model is processed with an algorithm. As a result, an ontology tree is constructed.

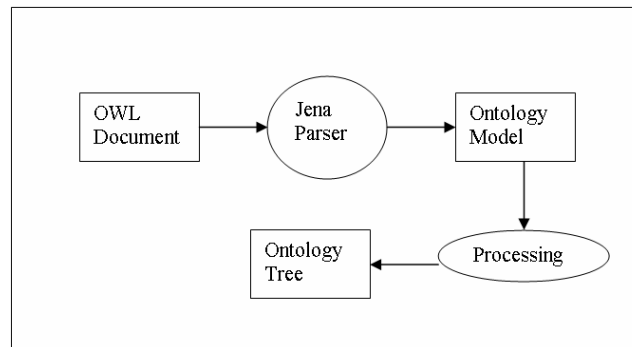


Figure 4.2 Constructing an ontology tree

Here is the processing algorithm pseudo-code:

```

createTree(OntologyModel model)

  for each class class1 of model

    if ( class1 has no super class )

      node1 = createNodes( all classes of model, class1 )

      add node1 into tree root

    return tree

createNodes(classlist1, class1)

node1 = Tree node ( class1 )

for each class class2 in classlist1

  if (class2 sub class of class1 )

    add createNodes ( classlist1, class2) under node1

  for each property property1 of class1

    addNodeIfDoesNotExist ( node1, property1 )
  
```

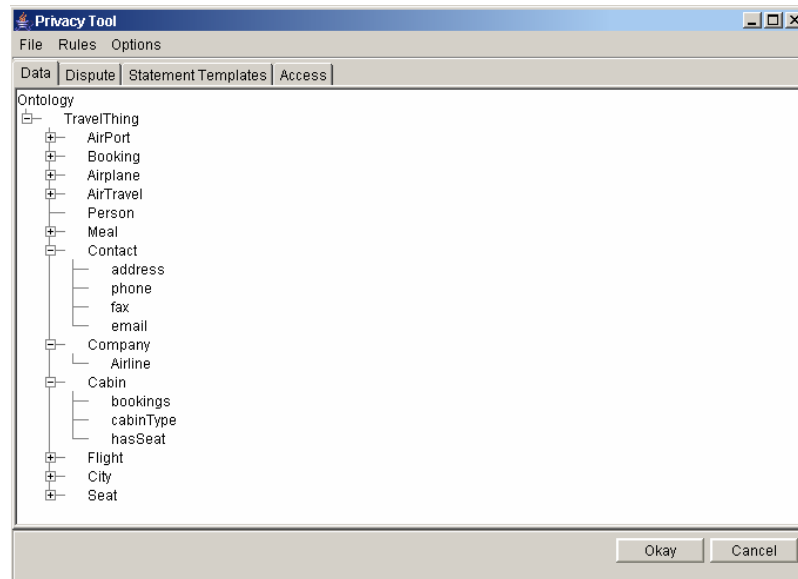


Figure 4.3 Ontology Tree

After constructing the ontology tree, the users select the program mode. Then they start to describe dispute resolution information.

The dispute resolution is another issue to be handled by our work. What to do when a disagreement about privacy issues occurs is clarified in this section. The potential methods are:

- **Correct:** Service management should undo the action that harms the user.
- **Money:** Service management pays a fine to the user that has been harmed by the service they offer.
- **Law:** The action which is to be performed will be determined by the law potentially stated in the document.

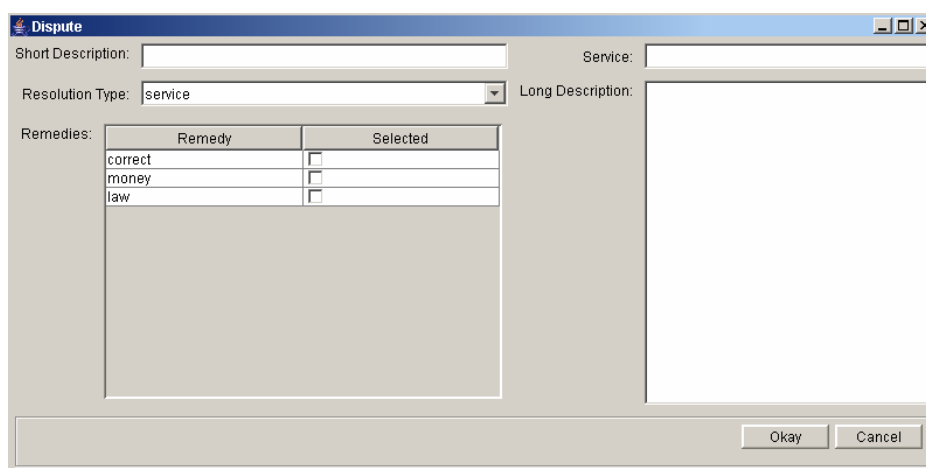
“Resolution Type” field describes which organization will dissolve the disputes. Possible values are:

- **Service:** A service that is used to resolve disputes
- **Independent:** An independent organization.
- **Court:** A specified court.
- **Law:** A specified law.

“Service” field is the URL of the resolution service.

The dispute resolution can be specified in the program as follows:

The user opens the ‘Dispute’ tab panel in the privacy frame. Then he/she clicks on the ‘Add Dispute’ button in order to see ‘Dispute’ frame. Users can specify the remedies for the disputes, resolution type, long description and verification via this frame.



Remedy	Selected
correct	<input type="checkbox"/>
money	<input type="checkbox"/>
law	<input type="checkbox"/>

Figure 4.4 Dispute frame

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

In the implementation, users are able to define the values for the fields above. The domain of values and the relationships have been fully adapted from P3P.

Since statements can be the same for different data nodes, in the thesis, statement template concept has been introduced. Statement templates can be defined in the ‘Statement Templates’ panel.

In the 'Statement Templates' panel, new statement templates can be defined, existing ones can be modified or deleted. In order to add a new statement template, 'Add Statement Template' button is clicked. It initiates the 'Statement Template' window.

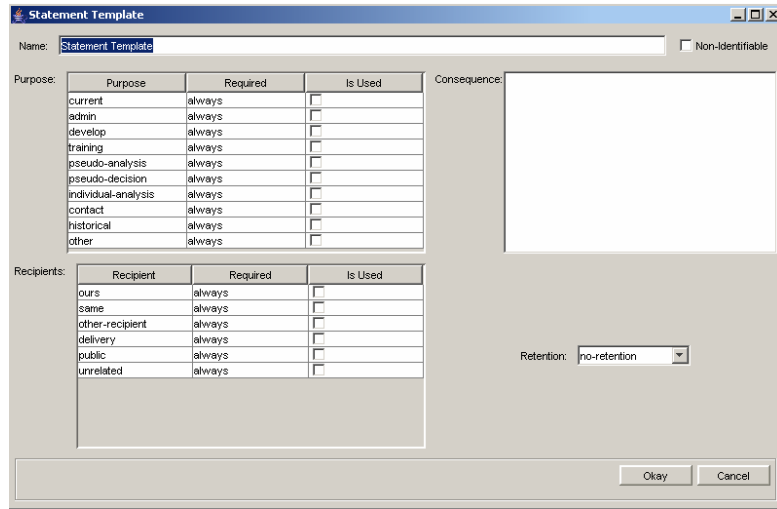


Figure 4.5 Statement template frame

In the window, the user can set the values for statement template fields described in Section 2.4.

After providing statement templates and disputes to the system, users can assign privacy preferences to the data extracted from the ontology. In order to achieve this, they open the first panel, 'Data' panel in the privacy frame. They select a data brunch from the ontology tree, then right-click on the tree. Prepared statement template can be assigned to the selected data by selecting the corresponding menu item under the 'Assign Statement Template' sub menu.

In the same panel, also policy preferences described in [1] can be defined. According to the mode of the system, service requestor or service provider choices can be selected from the 'Assign Privacy Value' menu.

After assigning privacy values, the user can save the privacy document. The resulting XML document is shown in aligned mode, and then it is saved in the file system. An example XML document is as following:

```
<?xml version="1.0" encoding="UTF-8"?>
<POLICY ontology="file:C:\tez\calisma\OTAHotelOntology_v1.rdfs"
workingmode="1">
  <ACCESS>contact-and-other</ACCESS>
```

```

<DISPUTES-GROUP>
  <DISPUTES resolution-type="service" service="http://service.org" short-
description="service">
    <LONG-DESCRIPTION>long description</LONG-DESCRIPTION>
    <REMEDIES>
      <correct/>
      <money/>
      <law/>
    </REMEDIES>
  </DISPUTES>
</DISPUTES-GROUP>
<STATEMENT name="Statement Template 1">
  <PURPOSE>
    <develop required="opt-in"/>
    <pseudo-Decision required="opt-in"/>
  </PURPOSE>
  <RECIPIENT>
    <same required="opt-in"/>
    <delivery required="opt-in"/>
  </RECIPIENT>
  <RETENTION>
    <stated-purpose/>
  </RETENTION>
  <CONSEQUENCE>Consequence</CONSEQUENCE>
  <DATA-GROUP>
    <DATA nodeType="2" ref="HotelThing.Person"/>
    <DATA nodeType="3" ref="HotelThing.Person.bookPerson"/>
    <DATA nodeType="1" ref="HotelThing.Person.name"/>
    <DATA nodeType="1" ref="HotelThing.Person.surname"/>
    <DATA nodeType="3" ref="HotelThing.RoomInfo"/>
  </DATA-GROUP>
</STATEMENT>
<RULES>
  <RULE/>
</RULES>
</POLICY>

```

4.2.4. Matching Documents

Two privacy documents (one provider and one requestor document) can be compared in the program. In the comparison, first, two documents are chosen from the file system. Then these documents are converted into the corresponding privacy policy objects. The ontology bases of these documents must be same so that they can match.

If that condition satisfies, then the dispute comparison is made. If any dispute statement of the first one is convenient with any dispute statement of the second one, dispute comparison results with a success.

For each dispute statement S1 of privacy document 1

For each dispute statement S2 of privacy document 2

If (isAppropriate(S1, S2))

Return true

Return false

The dispute statements are appropriate if resolution types are the same and remedy lists are appropriate.

isAppropriate(Dispute1, Dispute2)

if (resolutionType of Dispute1 is equal to resolution type of Dispute1

and isAppropriate(RemedyList1, RemedyList2))

return true;

return false

The remedy lists are appropriate, if there is at least one common remedy type in both lists.

isAppropriate(RemedyList1, RemedyList2)

for each remedy remedy1 of RemedyList1

for each remedy remedy2 of RemedyList2

if(remedy1 equals to remedy2)

return true

return false

For example, disputes in Figure 4.6 match, but ones in Figure 4.7 do not.

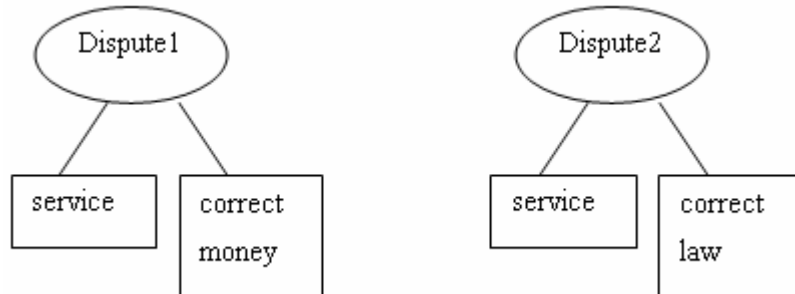


Figure 4.6 Disputes that match

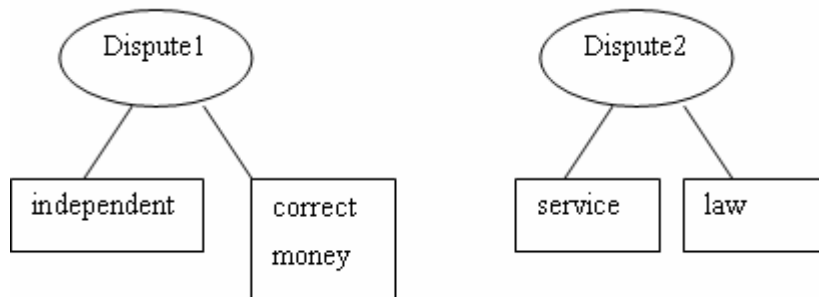


Figure 4.7 Disputes that do not match

After matching the dispute parts, the data parts are compared. First ontology trees are created from the privacy policy objects. Then policy values of the nodes of the trees are compared.

During creation of the trees, policy values of parent nodes are compared with the policy values of child nodes, more restricted values are assigned to child nodes. For example, if 'Address' node is 'Not Given' and its child node, 'Address.street', is 'Limited', then child node's new value will be 'Not Given'. The same procedure is applied to provider's privacy policy tree. Here are the precedence lists of policy values:

Not Given > Limited > Free

Mandatory > Optional

Then for each defined ontology node in provider's ontology tree, the corresponding node in requestor's ontology tree is found and their policy values are compared according to Table 4-1.

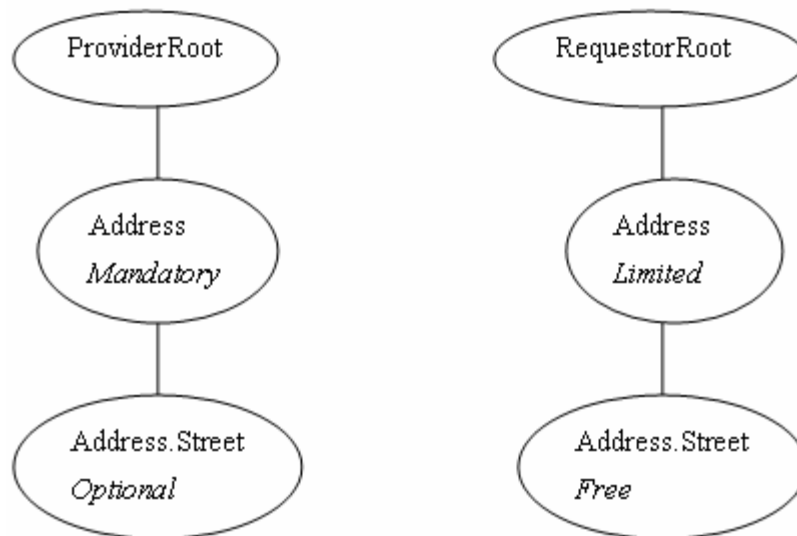


Figure 4.8 Ontology Trees

During negotiation phase, final value of 'Address.Street' becomes 'Mandatory' in the provider's ontology tree and 'Limited' in the requestor's ontology tree. While they seem to be suitable at a first glance, they do not match indeed.

In addition, the rules are taken into consideration in comparison. While looking up a node of provider's ontology tree, if there is a rule such that this node is the 'if part' of the rule, 'then part' of the rule is checked. Here is a new version of the previous example:

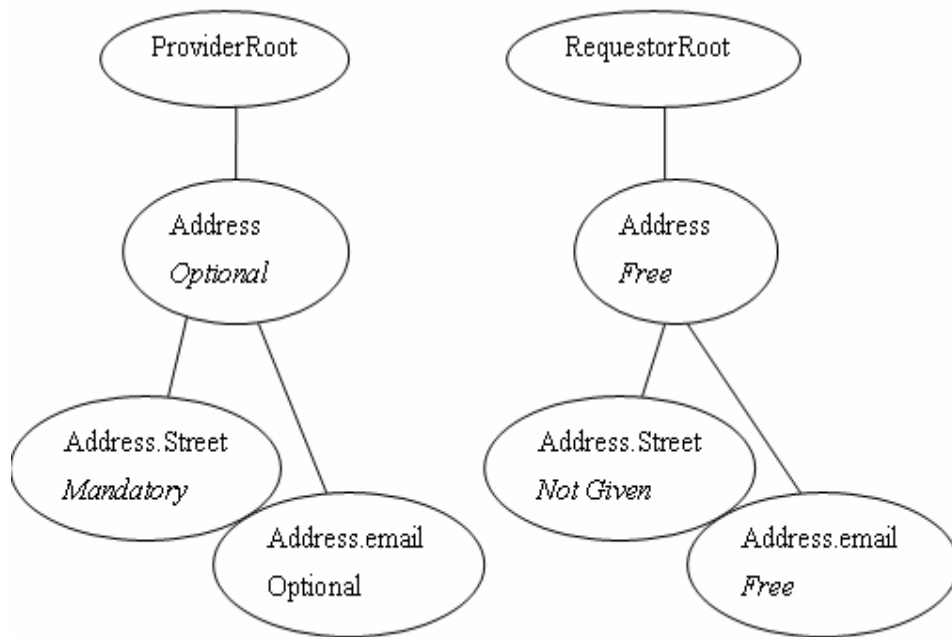


Figure 4.9 Ontology Trees 2

Rule: if Address.Street is 'Not Given', then Address.email is 'Mandatory'.

Without the rule, Address.Street nodes do not match, because it is a mandatory data for the provider but the requestor does not release it. Considering the rule, the 'if part' satisfies, because the Address.Street is 'Not Given'. With the rule, Address.email node becomes mandatory. Since the requestor releases Address.email freely, the nodes and trees match.

If policy values of nodes match, then privacy attributes are compared. For each node of provider's ontology tree, statement template of the node is extracted. This statement template is then compared with the statement template of the corresponding node of the requestor's tree. The comparison of statement templates includes comparisons of retention, recipient and purpose values. If retention values are not the same, ontology nodes do not match:

isAppropriate(Retention1, Retention2)

if(Retention1 is equal to Retention2)

return true

return false

If any value in provider recipient list does not exist in the requestor recipient list, then ontology nodes do not match.

isAppropriate(ProviderRecipientList, RequestorRecipientList)

for each recipient1 in ProviderRecipientList

if recipient1 is not in RequestorRecipientList

return false

return true

If any value in provider purpose list does not exist in the requestor purpose list, then ontology nodes do not match.

isAppropriate(ProviderPurposeList, RequestorPurposeList)

for each purpose1 in ProviderPurposeList

if purpose1 is not in RequestorPurposeList

return false

return true

If for each data node of provider ontology tree, there is a matching corresponding requestor node, then the privacy trees match.

For example, there are two ontology trees in the figure below.

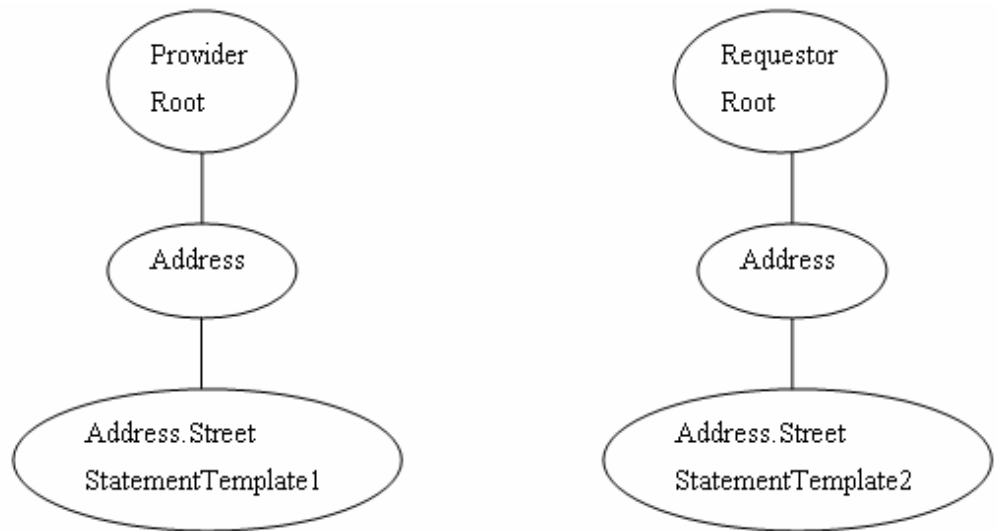


Figure 4.10 Trees with statement templates

Sample conditions are represented in Table 4.2.:

Table 4.2 Some statement template comparisons

Statement Template	Recipients	Retention	Purpose	Do Match? (Reason if not)
ProviderST	ours, same	no-retention	current, admin	Yes
RequestorST	ours	no-retention	admin	
ProviderST	public	no-retention	current, admin	No, recipients do not match
RequestorST	ours	no-retention	admin	
ProviderST	ours, same	stated-purpose	current, admin	No, retention policies do not match
RequestorST	ours	no-retention	admin	
ProviderST	ours, same	no-retention	current, admin	No, purposes do not match
RequestorST	ours	no-retention	training	

CHAPTER 5

5. CONCLUSION

A framework for web services privacy has been designed and implemented in the thesis. It uses message ontologies in order to provide a basis for the privacy documents. Upon those ontologies, privacy preferences about the web service providers and the web service requestors can be defined within a graphical user interface. This interface allows users to express the privacy preferences according to the web services privacy specification as given in [2] and according to the work described in [1]. The privacy documents are saved in eXtensible Markup Language (XML) format. In case those documents are deployed in a 'proper' place, they can be used as the criteria to match the privacy preferences of the provider and the requestor. After creation of these documents, they can be edited in the graphical user interface.

The cornerstone of the work is creating privacy documents easily, being able to edit them and comparing those documents. There is no need to define privacy documents by writing XML documents. Also web services can determine automatically whether the service requestor accepts the privacy preferences.

5.1. Work Realized

In the thesis, a web service privacy tool has been designed and implemented. The requirements of the implementation are gathered from the specification [2] and the paper [1]. In the design phase, modeling was realized by drawing UML diagrams. The model architecture tried to respond all the requirements extracted in the previous step. Java programming language has been used as the development platform. Java Swing library has been sufficient for the graphical user interface development.

Privacy documents can be easily created and modified with the help of software product of the thesis. Also a matching algorithm has been devised and implemented as a part of the work. Privacy documents can be compared in the tool, but it will be better to match those documents during the web service operation.

5.2. Related Work

W3C provides a language called P3P Preference Exchange Language 1.0 (APPEL1.0). APPEL is proposed to be used to express users' preferences for making automated or semi-automated decisions regarding the acceptability of machine-readable privacy policies from P3P enabled Web sites [36].

AT&T developed a P3P user agent, Privacy Bird [38]. It is integrated with the Microsoft Internet Explorer 5.1 and later versions. It checks whether a web site's P3P policy matches a user's privacy preferences.

The work described in [35] proposes anonymous group identification for privacy issues in web services. It enlarges SOAP packets in order to enable web services to understand the groups of users without fetching their identification information.

Enterprise Privacy Authorization Language (EPAL) technical specification produced by IBM is used to formalize privacy authorization for actual enforcement within an intra- or inter- enterprise for business-to-business privacy control [37]. EPAL services themselves are exchanging privacy policies and making privacy authorization decisions. In particular, EPAL concentrates on the privacy authorization by abstracting data models and user authentication from all deployment details. However, the EPAL framework does not consider the privacy enforcement in the context of WSA.

5.3. Comments

The thesis work tries to fill up a gap, which resides in using the web service privacy concept. Privacy implementations have been developed for web sites up to now. This implementation tries to construct web service privacy documents that can be directly used by the web services and then develop a framework for the web service privacy. Also this implementation tries to enhance the privacy specification with the existence rules defined in [1]. It will enable users to prepare more flexible privacy documents.

The performance of the work has not been a problem so far because toy ontologies are used as the data sources. The performance issue should be investigated with larger ontologies.

5.4. Future Work

As the future work, the framework implementation should continue with providing a matching mechanism just before the web services are invoked. Also privacy document deployment to a 'known' and 'proper' location remains as a to-do job. Also the performance of the work should be examined with larger message ontologies.

REFERENCES

- [1] Arif Tumer, Asuman Dogac, and I. Hakki Toroslu, "A Semantic-based User Privacy Protection Framework for Web services", Software Research and Development Center & Dept. of Computer Eng., Middle East Technical University
- [2] The Platform for Privacy Preferences 1.0 (P3P1.0) Specification, W3C Recommendation, 16 April 2002, <http://www.w3.org/P3P>
- [3] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web", *Scientific American*, 279(5):34.43, May 2001
- [4] Paul Ashley, Calvin Powers, Matthias Schunter, "From Privacy Promises to Privacy Management - A New Approach for Enforcing Privacy Throughout an Enterprise", New Security Paradigms Workshop '02, 2002
- [5] Abdelmounaam Rezgui, Mourad Ouzzani, Athman Bouguettaya, Brahim Medjahed, "Preserving Privacy in Web Services", WIDM'02, November 8, 2002
- [6] Lorrie Faith Cranor, "The Role of Privacy Advocates and Data Protection Authorities in the Design and Deployment of the Platform for Privacy Preferences", 1-6, *Communications of the ACM*
- [7] Lorrie Faith Cranor, "Internet Privacy", *Communications of the ACM*, February 1999/Vol. 42, No. 2
- [8] Mark S. Ackerman, Lorrie Faith Cranor, Joseph Ragle, "Privacy in E-Commerce: Examining User Scenarios and Privacy Preferences", 1999
- [9] Workshop on Consumer Privacy on the Global Information Infrastructure. <http://www.ftc.gov/bcp/privacy/wkshp96/privacy.htm>
- [10] FTC Comment: Script of W3C P3 Prototype. <http://www.w3.org/Talks/970612-ftc/ftc-sub.html>
- [11] Removing Data Transfer from P3P. <http://www.w3.org/P3P/data-transfer.html>

- [12] Aaron Goldfeder, Lisa Leibfried, "Privacy in Internet Explorer 6", <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnpriv/html/ie6privacyfeature.asp>.
- [13] Kaler, C., and Nadalin, A. Web Services Trust Language (WS-Trust). IBM and Microsoft, December 2002, <http://www.ibm.com/developerworks/library/wstrust/>.
- [14] Children's Online Privacy Protection Act of 1998 (COPPA), October 1998. Available at www.cdt.org/legislation/105th/privacy/coppa.html
- [15] Lorrie Faith Cranor, "Privacy Tools Overview", September 2000, <http://lorrie.cranor.org/pubs/privacy-tools-sept2000.html>
- [16] Joseph Reagle, Lorrie Faith Cranor, "The Platform for Privacy Preferences", Communications of the ACM, February 1999/Vol. 42, No. 2
- [17] Heather Kreger, "Fulfilling the Web Services Promise", Communications of the ACM, June 2003/Vol. 46, No. 6
- [18] Hristo Koshutanski, Fabio Massacci, "An Access Control Framework for Business Processes for Web Services", ACM Workshop on XML Security, October 31, 2003
- [19] OWL Web Ontology Language Reference, W3C Recommendation, 10/02/2004 <http://www.w3.org/TR/owl-ref/>
- [20] OWL Web Ontology Language Overview, W3C Recommendation, 10/02/2004 <http://www.w3.org/TR/owl-features/>
- [21] Jena Project, A Semantic Web Framework for Java, <http://jena.sourceforge.net/>
- [22] HIPAA States, The Health Insurance Portability and Accountability Act of 1996 (HIPAA), October 1998, <http://www.hcfa.gov/hipaa/hipaahm.html>
- [23] United States, Gramm-Leach-Bliley Act: Financial Privacy and Pretexting, November 1999, <http://www.ftc.gov/privacy/glbact/glboutline/htm>
- [24] Forrester Research, "Privacy Concerns Cost e-Commerce \$15 Billion", September 2001, <http://www.forrester.com>
- [25] Marian Ventuneac, Tom Coffey, Ioan Salomie, "A Policy-Based Security Framework for Web-Enabled Applications", University of Limerick

[26] Anne H. Anderson, “An Introduction to the Web Services Policy Language (WSPL)”, Proceedings of the Fifth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY’04)

[27] Web Services Policy Framework (WS-Policy),
<http://www106.ibm.com/developerworks/library/ws-polfram/>, May 2003.

[28] Dan Hong, Mingxuan Yuan, Vincent Y. Chen, “Dynamic Privacy Management: a Plugin Service for the Middleware in Pervasive Computing”, MobileHCI’05, September 19–22, 2005, Salzburg, Austria

[29] Unified Communications Glossary of Terms, Communications 4 Network Services, Berkeley University of California, <http://unibears.berkeley.edu/glossary.html>

[30] Venu Vaseduvan, “A Web Services Primer”, April 04, 2001,
<http://webservices.xml.com/pub/a/ws/2001/04/04/webservices/>

[31] Bellman, “Web Services Standards”, May 15, 2002,
<http://www.it-director.com/article.php?id=2840>

[32] David Orchard, “Making Sense of Web Services Standards”, January 30, 2004,
http://dev2dev.bea.com/pub/a/2004/01/ws_orchard.html

[33] SATINE Project Web Site,
<http://www.srdc.metu.edu.tr/webpage/projects/satine/>

[34] A. Doğaç, Y. Kabak, G. Lalaeci, S. Sınır, A. Yıldız, A. Tümer, “SATINE Project: Exploiting Web Services in the Travel Industry”, 2004, METU, Ankara

[35] Giuseppe Cattaneo, Pompeo Faruolo, Umberto Ferraro Petrillo, Giuseppe Persiano, “Providing Privacy for Web Services by Anonymous Group Identification”, Proceedings of the IEEE International Conference on Web Services, 2004

[36] World Wide Web Consortium (W3C). “A P3P Preference Exchange Language 1.0 (APPEL1.0),” W3C Working Draft, April 15, 2002,
<http://www.w3.org/TR/P3P-preferences/>

[37] IBM Corporation, “Enterprise Privacy Authorization Language (EPAL)”, IBM Research Report, 2003,
<http://www.zurich.ibm.com/security/enterprise-privacy/epal>

[38] Lorrie Faith Cranor, Manjula Arjula, Praveen Guduru, “Use of a P3P User Agent by Early Adopters”, ACM Digital Library

[39] The Open Travel Alliance Web Site, <http://www.opentravel.org>

[40] Unified Modeling Language Web Site, <http://www.uml.org>

[41] Rational XDE Developer Java Web Site,
<http://www-306.ibm.com/software/awdtools/developer/java>

[42] Java Programming Language Web Site, <http://java.sun.com>

[43] Resource Description Framework (RDF) Web Site, <http://www.w3.org/RDF>

APPENDIX A

A. WEB SERVICE PRIVACY FRAMEWORK DESIGN

A.1. Class Diagrams

Packages

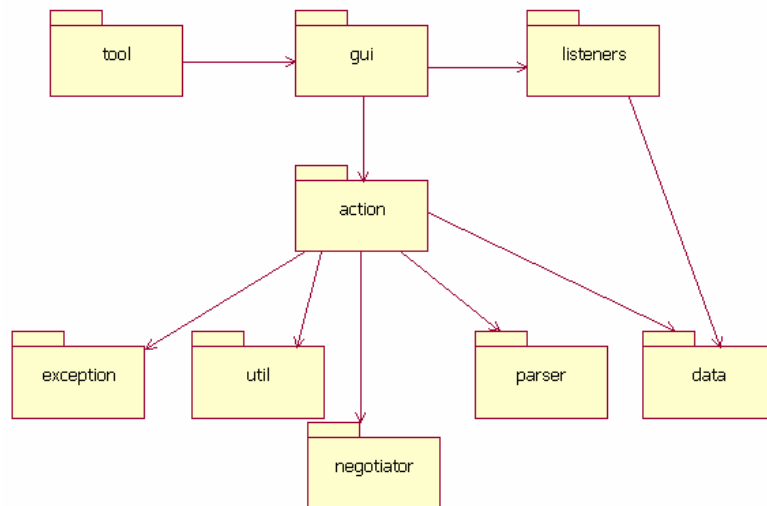


Figure A.1 Packages

tool package

This package contains the main class of the program, PrivacyTool class. This package is used to initiate the program, selecting the main actions such as creating or modifying a privacy document.

PrivacyTool Class

This class is the main class of the whole program. It initiates the program, shows the main frame, and holds the methods that initiate the creation and modification of the privacy documents.

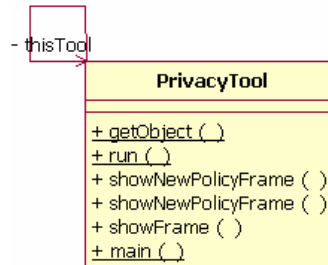


Figure A.2 PrivacyTool Class

Fields

private static PrivacyTool thisTool

This field is used to ensure that only one privacy tool object exists throughout the program. Methods of PrivacyTool class are accessed through this field.

Methods

public static PrivacyTool getObject()

This method creates one PrivacyTool object if it does not exist, and gives the reference to the other classes. This method is also used to ensure that only one privacy tool object exists throughout the program. Methods of the PrivacyTool class are called via this method, for instance, PrivacyTool.getObject().showFrame().

public static void main(String[] args)

This method initiates the whole program. The software is started to run by the PrivacyTool class, by the default call of the Java virtual machine to this method.

args parameter represents the command line parameters to the method.

public static void showFrame ()

This method shows the main frame of the program. After the software starts, this frame is shown.

public static void showNewPolicyFrame ()

This method shows a new privacy frame in the main frame of the program. After the frame is shown, the user can define a new privacy policy document.

public static void showNewPolicyFrame (Policy policy)

This method shows a new privacy frame in the main frame of the program. The privacy frame is used to modify an existing privacy policy document.

policy parameter is the privacy policy object that represents the privacy document to be modified.

public static void run ()

This method is called in the main method. It initiates the program.

gui.access package

This package contains the necessary user interface to handle *access* attribute of the privacy policy.

AccessPanel Class

This class constitutes the panel that can enable user to select access value of the privacy document.

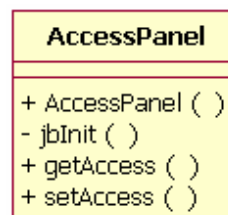


Figure A.3 AccessPanel Class

Fields

None

Methods

public AccessPanel()

This method is the public constructor of the `AccessPanel` class.

private void jbInit()

This method initiates the panel, i.e. adds the panel content into the panel.

private void getAccess ()

This method gets the access value that has been selected by the user from the access panel.

private void setAccess (String access)

This method shows the access value of the privacy policy, which is to be modified, in the access panel.

access parameter is the access value of the privacy policy document, which is modified.

gui.comparison package

This package contains the necessary user interface to handle comparison of the privacy policies. The comparison is realized in a separate frame that enables users to select privacy documents.

ComparisonFrame Class

This class constitutes the frame that can enable users to compare two privacy documents. This frame holds two buttons to select the service provider and the service requestor privacy documents which shall be compared. The third button initiates the comparison. The result of the comparison can be seen in a dialog.

ComparisonFrame
+ ComparisonFrame ()
- init ()
+ actionPerformed ()
- btnOkayPressed ()
- selectPolicyFileName ()
- selectPolicy1 ()
- selectPolicy2 ()

Figure A.4 ComparisonFrame Class

Fields

None

Methods

public ComparisonFrame()

This method is the public constructor of the ComparisonFrame class.

private void init()

This method initiates the frame, i.e. adds the frame content (buttons, labels ...) into the frame.

public void actionPerformed()

This method sets the actions of the buttons.

private void btnOkayPressed()

This method is called when the “Okay” button is pressed. It starts the comparison process.

private String selectPolicyFileName ()

This method shows a file dialog. If the user selects a file and approves the selection, this method returns the full path of the selected file. This method is used by the *selectPolicy1()* and *selectPolicy2()* methods.

private void selectPolicy1 ()

Using the *selectPolicyFileName()*, it shows the selected file name in the “Provider Policy” text field. It sets the service provider privacy document which will be compared.

private void selectPolicy2 ()

Using the *selectPolicyFileName()*, it shows the selected file name in the “Requestor Policy” text field. It sets the service requestor privacy document which will be compared.

gui.datapanel package

This package contains the necessary user interface to show the ontology as a tree and assign privacy and policy content to the data in that ontology. After the selection of the base

ontology, its content is drawn as a data tree on the data panel. With a right click on a node of the tree, a popup menu is shown. This popup menu enables users to select the privacy and policy content for the selected node.

DataPanel Class

This is an important class, which shows the content of the ontology. It also enables users to assign privacy and policy attributes to the data nodes of the ontology.

DataPanel
+ setTree () + DataPanel () + assignData () + populateStatements () - buildPopupMenu () + setStatementList () - assignToRule () - createRule ()

Figure A.5 DataPanel Class

Fields

None

Methods

public DataPanel()

This method is the public constructor of the DataPanel class.

public void setTree(JTree tree)

This method shows the tree in the data panel.

public void assignData()

This method assigns the statement attributes to the data items on the ontology tree.

public ArrayList populateStatements()

This method extracts the selected data items from the ontology tree, and then constructs a proper object list that represents data and their privacy and policy attributes.

private String buildPopupMenu ()

This method constructs popup menus for the ontology nodes. If the system has at least one statement, the menu contains “Assign Statement Template” menu item.

public void setStatementList (ArrayList statementList)

While loading a privacy document, this method assigns statement values to the ontology tree nodes in order to show in the panel.

statementList parameter is the list of statements that will be shown on the ontology tree.

private void createRule (OntologyNode node)

Creates a new policy rule and assigns *node* to the ‘if’ part of the rule.

node parameter is the ‘if’ part of the new rule.

private void assignToRule (OntologyNode node)

This method sets the ‘then’ part of the rule as ‘node’.

node parameter is the ‘then’ part of the new rule.

gui.dispute package

This package contains the necessary user interface to handle dispute management. Disputes can be defined, erased, and modified in this package.

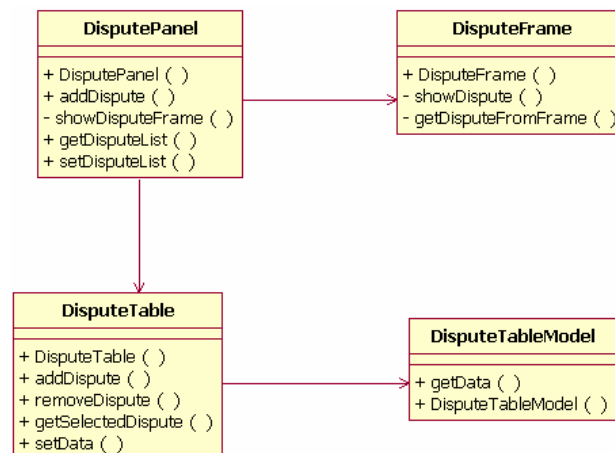


Figure A.6 Dispute package

DisputePanel Class

This panel shows the dispute list of the privacy document on the privacy frame. It can also start the addition of a new dispute or modify an existing one. It can also delete a dispute from the privacy document.

Fields

None

Methods

public DisputePanel()

This method is the public constructor of the DisputePanel class.

public void addDispute(Dispute dispute)

This method adds a new dispute object into the dispute table on the panel.

private void showDisputeFrame()

This method opens the dispute frame in order to add a new dispute or modify an existing one.

public ArrayList getDisputeList()

This method returns the disputes listed on the dispute table.

public void setDisputeList(ArrayList disputeList)

This method shows the dispute list on the dispute table.

DisputeFrame Class

This frame enables users to describe a new dispute or modify an existing one.

Fields

None

Methods

public DisputeFrame()

This method is the public constructor of the DisputeFrame class.

private void showDispute(Dispute dispute)

This method shows the dispute to be modified.

private Dispute getDisputeFromFrame()

This method creates a dispute object and sets the fields of it with the information provided on the frame.

DisputeTable Class

This table lists the disputes on the dispute panel list.

Fields

None

Methods

public DisputeTable()

This method is the public constructor of the DisputeTable class.

public void addDispute(Dispute dispute)

This method adds the dispute into the dispute list and shows in the table.

public void removeDispute(Dispute dispute)

This method removes the dispute from the dispute list.

public Dispute getSelectedDispute()

This method returns the selected (if there is a selected one) dispute in the table. If no dispute is selected, then returns *null*.

public void setData(ArrayList disputeList)

This method adds all the disputes in the list into the table.

DisputeTableModel Class

This class is the table model of the DisputeTable class.

Fields

None

Methods

public DisputeTableModel()

This method is the public constructor of the DisputeTableModel class.

public ArrayList getData ()

This method returns the dispute list of the dispute table.

gui.mainframe package

This package contains the class of the main frame of the program.

MainFrame Class

This class constitutes the main frame of the program.

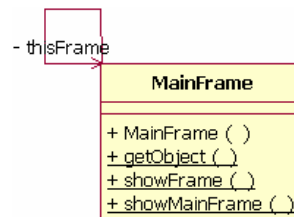


Figure A.7 MainFrame Class

Fields

private static MainFrame thisFrame

This field is the unique occurrence of the MainFrame object throughout the program. Methods of MainFrame class are accessed through this field.

Methods

public MainFrame()

This method is the public constructor of the MainFrame class.

public static MainFrame getObject(JTree tree)

This method is used to ensure that only one MainFrame object exists throughout the program. It returns the unique MainFrame object. Methods of *MainFrame* class are accessed through this method.

public static void showMainFrame()

This method shows the unique main frame, if it does not exist, creates it.

public static void showFrame(ErkanarInternalFrame frame)

This method shows the internal frame in the main frame.

gui.privacy package

This package contains the class of the main frame of the program.

PrivacyFrame Class

This class constitutes the main frame of the program.

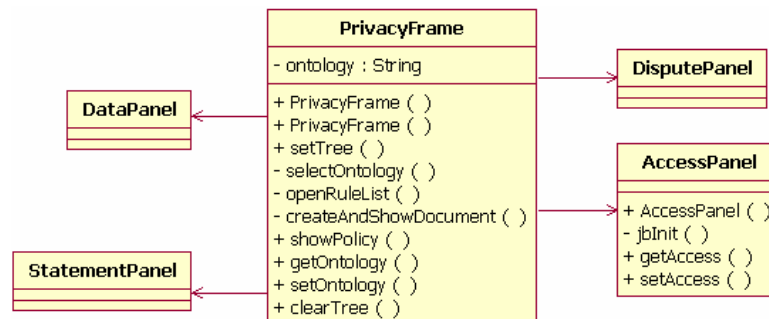


Figure A.8 PrivacyFrame Class

Fields

private String ontology

This field is the URL of the ontology document that will be shown as a tree.

Methods

public PrivacyFrame()

This method is the public constructor of the PrivacyFrame class.

public PrivacyFrame (Policy policy)

This method is the public constructor of the PrivacyFrame class. It shows a privacy document's content in order to enable users to modify it.

public void setTree (JTree tree)

This method shows the tree.

private void selectOntology ()

This method opens a file dialog and enables users to select an ontology document (an OWL document) .

private void openRuleList ()

This method opens the rule list of the privacy document.

private void openRuleList ()

This method opens the rule list of the privacy document.

private void createAndShowDocument ()

This method creates a privacy policy document with the information the user has been provided. It also shows the produced document in indented XML format.

public void showPolicy (Policy policy)

This method shows a privacy policy document's content in order to enable users to modify it.

public String getOntology ()

This method returns the URL of the ontology document.

public String setOntology (String ontology)

This method sets the ontology document of the privacy frame. It also shows the ontology as a tree.

public void clearTree ()

This method clears the policy attributes of the nodes of the ontology tree.

gui.purpose package

This package contains the necessary user interface to handle purpose attributes of statements.

PnlPurpose Class

This panel shows the purpose list of a statement of the privacy document on the statement frame. It can also change the purpose attributes of the statement.

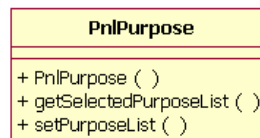


Figure A.9 PnlPurpose Class

Fields

None

Methods

public PnlPurpose()

This method is the public constructor of the PnlPurpose class.

public void setPurposeList (ArrayList purposeList)

This method sets the purpose attributes of the panel.

public ArrayList getPurposeList ()

This method returns the purpose attributes of the panel.

PurposeTable Class

This table lists purpose attributes of the statement. On the table, these attributes can also be changed.

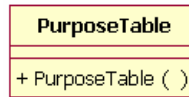


Figure A.10 PurposeTable Class

Fields

None

Methods

public PurposeTable ()

This method is the public constructor of the PurposeTable class.

PurposeTableModel Class

This class is the table model of the PurposeTable class.

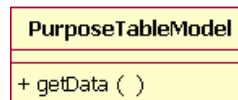


Figure A.11 PurposeTableModel Class

Fields

None

Methods

public PurposeTableModel()

This method is the public constructor of the PurposeTableModel class.

public ArrayList getData ()

This method returns the purpose list of the purpose table.

ConsequencePanel Class

This class is the table model of the ConsequencePanel class.

ConsequencePanel
+ ConsequencePanel ()
+ getConsequence ()
+ setConsequence ()

Figure A.12 ConsequencePanel Class

Fields

None

Methods

public ConsequencePanel ()

This method is the public constructor of the ConsequencePanel class.

public Consequence getConsequence ()

This method returns the consequence attribute of the statement from the statement frame.

public void setConsequence (Consequence consequence)

This method sets the consequence value in the statement frame.

gui.recipient package

This package contains the necessary user interface to handle recipient attributes of statements.

RecipientPanel Class

This panel shows the recipient list of a statement of the privacy document on the statement frame. It can also change the recipient attributes of the statement.

RecipientPanel
+ RecipientPanel ()
+ getSelectedRecipientList ()
+ setRecipientList ()

Figure A.13 RecipientPanel Class

Fields

None

Methods

public RecipientPanel()

This method is the public constructor of the RecipientPanel class.

public void setRecipientList (ArrayList recipientList)

This method sets the recipient attributes of the panel.

public ArrayList getSelectedRecipientList ()

This method returns the recipient attributes of the panel.

RecipientTable Class

This table lists recipient attributes of the statement. On the table, these attributes can also be changed.

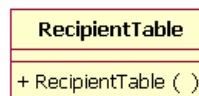


Figure A.14 RecipientTable Class

Fields

None

Methods

public RecipientTable ()

This method is the public constructor of the RecipientTable class.

RecipientTableModel Class

This class is the table model of the RecipientTable class.

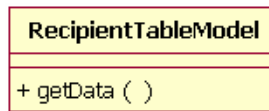


Figure A.15 RecipientTableModel Class

Fields

None

Methods

public RecipientTableModel ()

This method is the public constructor of the RecipientTableModel class.

public ArrayList getData ()

This method returns the recipient list of the table.

gui.remedy package

This package contains the necessary user interface to handle remedy attributes of statements.

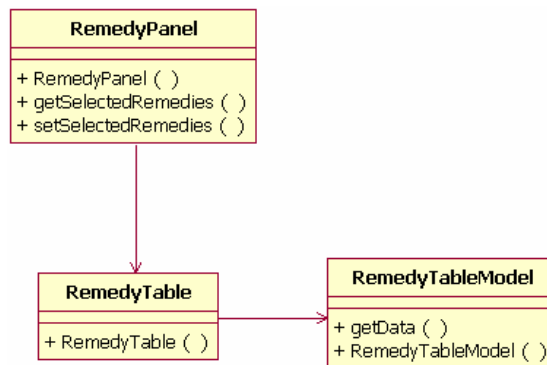


Figure A.16 gui.remedy package

RemedyPanel Class

This panel shows the remedy list of a statement of the privacy document on the statement frame. It can also change the remedy attributes of the statement.

Fields

None

Methods

public RemedyPanel()

This method is the public constructor of the RemedyPanel class.

public void setSelectedRemedies (ArrayList remedyList)

This method sets the selected remedy attributes of the panel.

public ArrayList getSelectedRemedies ()

This method returns the selected remedy attributes of the panel.

RemedyTable Class

This table lists remedy attributes of the statement. On the table, these attributes can also be changed.

Fields

None

Methods

public RemedyTable ()

This method is the public constructor of the RemedyTable class.

RemedyTableModel Class

This class is the table model of the RemedyTable class.

Fields

None

Methods

public RemedyTableModel ()

This method is the public constructor of the RemedyTableModel class.

public ArrayList getData ()

This method returns the remedy list of the table.

gui.retention package

This package contains the necessary user interface to handle retention attributes of statements.

RetentionPanel Class

This panel shows the retention value of a statement of the privacy document on the statement frame. It can also change the retention attribute of the statement.

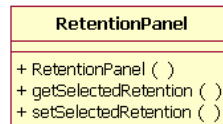


Figure A.17 RetentionPanel class

Fields

None

Methods

public RetentionPanel ()

This method is the public constructor of the RetentionPanel class.

public void setSelectedRetention (Retention retention)

This method sets the selected retention attribute of the panel.

public ArrayList getSelectedRetention ()

This method returns the selected retention attribute of the panel.

gui.rule package

This package contains the necessary user interface to handle retention attributes of statements.

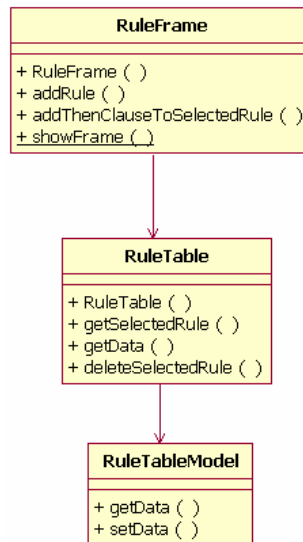


Figure A.18 gui.rule package

RuleFrame Class

This frame shows the policy rule list of a privacy document. It can also enable users to modify the policy rules of the privacy document.

Fields

None

Methods

public RuleFrame ()

This method is the public constructor of the RuleFrame class.

public void addRule (Rule rule)

This method adds a new rule (only if part) to the rule list in the frame.

public void addThenClauseToSelectedRule (OntologyNode thenPart)

This method adds the then part of the rule to the selected rule in the frame.

public static void showFrame (Policy policy, boolean refreshTable)

This method shows a rule frame for the given policy. *refreshTable* parameter states whether the rule list on the frame will be refreshed.

RuleTable Class

This table lists rules of the privacy document. On the table, these rules can also be changed or deleted.

Fields

None

Methods

public RuleTable ()

This method is the public constructor of the RuleTable class.

public Rule getSelectedRule ()

This method returns the selected rule in the rule table.

public ArrayList getData ()

This method returns all the rules in the rule table.

public void deleteSelectedRule ()

This method deletes the selected rule in the rule table.

RuleTableModel Class

This class is the table model of the RuleTable class.

Fields

None

Methods

public ArrayList getData ()

This method returns the rule list of the table.

public void setData (ArrayList data)

This method sets the rules of the table.

gui.statement package

This package contains the necessary user interface to handle statements of privacy document.

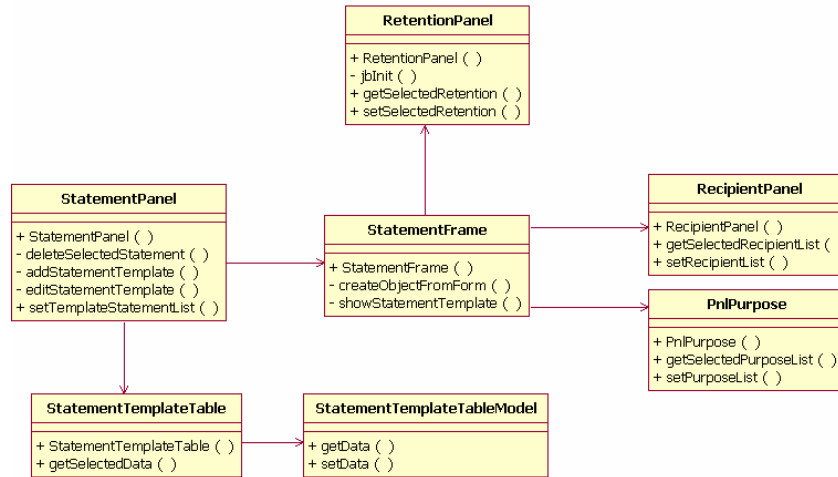


Figure A.19 gui.statement package and relationships with other panels

StatementFrame Class

This frame enables users to define a new statement template or modify an existing one.

Fields

None

Methods

public StatementFrame (Policy policy, StatementTemplate statementTemplate, StatementPanel statementPanel)

This method is the public constructor of the StatementFrame class.

policy parameter is the privacy policy, for which a new statement template is defined.

statementTemplate parameter is the statement template whose data will be changed. If the frame is shown to describe a new template, then this parameter is given *null*.

statementPanel parameter is the statement panel which has opened the frame. Newly defined statement template will be added into this panel.

private StatementTemplate createObjectFromForm ()

This method sets the fields of the statement template with the information provided by the user and returns it.

private void showStatementTemplate (StatementTemplate st)

This method shows the fields of the statement template on the frame.

StatementPanel Class

This panel lists the statement templates defined by the user.

Fields

None

Methods

public StatementPanel ()

This method is the public constructor of the StatementPanel class.

private void deleteSelectedStatement ()

This method deletes the selected statement template from the table.

private void addStatementTemplate()

This method opens a StatementFrame frame and initiates to describe a new statement template.

private void editStatementTemplate()

This method opens a StatementFrame frame and initiates to edit the selected statement template in the table.

public void setTemplateStatementList(ArrayList statementTemplateList)

This method adds all the statements in the list into the statement template table.

StatementTemplateTable Class

This table lists statement templates of the privacy document.

Fields

None

Methods

public StatementTemplateTable ()

This method is the public constructor of the StatementTemplateTable class.

public ArrayList getSelectedData ()

This method returns the selected statement template of the table.

StatementTemplateTableModel Class

This class is the table model of the StatementTemplateTable class.

Fields

None

Methods

public ArrayList getData ()

This method returns the statement template list of the table.

public void setData (ArrayList data)

This method sets the statement templates of the table.

gui.xml package

This package contains the necessary user interface to show the privacy documents in XML format.

XMLFrame Class

This frame enables users to see the privacy document in XML format.

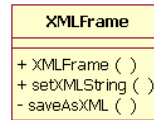


Figure A.20 XMLFrame class

Fields

None

Methods

public XMLFrame (String xmlString)

This method is the public constructor of the XMLFrame class.

xmlString parameter is the privacy document representation in XML format.

public void setXMLString (String xml)

This method shows the given string in the frame.

xml parameter is the string that will be shown in the frame.

public void saveAsXml ()

This method opens a file dialog and saves the XML string as the selected file.

util package

This package contains the utility classes of the program.

XMLDocumentCreator Class

This class converts privacy policy objects into the XML document objects.

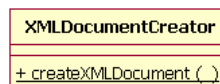


Figure A.21 XMLDocumentCreator class

Fields

None

Methods

public Document createXMLDocument (Policy policy)

This method converts the given privacy policy into an XML document.

policy parameter is the privacy document, which will be converted into an XML document.

parser package

This package contains the parser classes of the program.

PrivacyParser Class

This class reads privacy documents and converts them into privacy policy objects.



Figure A.22 PrivacyParser class

Fields

None

Methods

public static Policy processPrivacyFile (String fileName)

This method converts the file with the given path into a privacy policy object.

fileName is the path of the privacy policy file, which will be converted into a privacy policy object.

public static Document parseDocument (String fileName)

This method converts the file with the given path into an XML document.

fileName is the path of the privacy policy file, which will be converted into an XML document.