DIP
Data, Information and Process Integration with Semantic Web Services
FP6 - 507483

Deliverable

WP 3: Service Ontology and Service Description
D3.6
Trading partner management and trust

Stefania Galizia
Andreas Friesen

January 20th, 2006
SUMMARY

The objective of work-package 3 is to employ the ontology and Semantic Web infrastructure by adding semantics to the Web Service description. This deliverable presents a general framework that fulfils given requirements for trading partner management and trust properties of Semantic Web. We provide an ontology, named \textit{WSTO} (Web Service Trust-management Ontology), through which both requester and Web Service provider can express their individual policies. Then, reasoning modules will activate trust-based Semantic Web Service selection. We characterise trust analysis as a classification process, within which valid solutions are those Web Services that match given classification criteria. The proposed model is very general, in order to accommodate extensions to a variety of preferences and requirements of entities taking part in a conversation.

According to main goals of DIP, one of the contributions of this document will be an implementation within IRS-III, a DIP architecture and tool.

This work is related to the deliverables D4.17 “SWS Discovery Module Specification & P2P QoS-enabled Discovery Specification” and D4.8 “Discovery Specification”. The trading partner management and trust issues could make use of QoS-notion described in D4.17 and mentioned in D4.8; moreover as D4.17, this deliverable proposes a WSMO extension.

However, while D4.17 and D4.8 deal with QoS-based and Semantic Web Service discovery, we investigate on a general ontological framework allowing to describe and evaluate trust- and security-related requirements in order to support service selection taking into account those requirements.

The intended target audience of this deliverable is anybody who is interested in modelling trading partner information, and trust requirements and criteria in Semantic Web Services environment.

Serving as the specification for describing trust-related aspects of Semantic Web Services within the DIP project, this document is expected to be used in the deliverable D4.21, “Trust-based Invocation Tool” and within work package 9 in the tasks “SWS Enhanced GIS Prototype”, version “v1.0” and/or “v2.0”.

Disclaimer: The DIP Consortium is proprietary. There is no warranty for the accuracy or completeness of the information, text, graphics, links or other items contained within this material. This document represents the common view of the consortium and does not necessarily reflect the view of the individual partners.
Deliverable 3.6

Document Information

<table>
<thead>
<tr>
<th>IST Project Number</th>
<th>FP6 – 507483</th>
<th>Acronym</th>
<th>DIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full title</td>
<td>Data, Information, and Process Integration with Semantic Web Services</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project URL</td>
<td><a href="http://dip.semanticweb.org">http://dip.semanticweb.org</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Document URL</td>
<td>Kai Tullius</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Deliverable Number 3.6 | Title | Trading partner management and trust
Work package Number 3  | Title | Ontology and Service Description

Date of delivery Contractual 24 Actual 31-Dec-05
Status version. 1.0 final
Nature Prototype Report Dissemination Ontology
Dissemination Level Public Consortium

Authors (Partner) Stefania Galizia (The Open University), Andreas Friesen (SAP)
Responsible Author Stefania Galizia
Partner The Open University
Email s.galizia@open.ac.uk
Phone +44 (0) 1908 659461

Abstract (for dissemination) The objective of work-package 3 will be to employ the ontology and Semantic Web infrastructure by adding semantics to the web service description. This deliverable provides a framework that allows all partners, involved in a business transaction, to manage their information and policies. We focus especially on trust policies in Semantic Web Services environment, by facilitating both user and Web Service to apply their individual trust policies and enabling a trust-based selection process.

Keywords Trust, Security, Ontologies, Semantic Web Services, SWS selection

Version Log

<table>
<thead>
<tr>
<th>Issue Date</th>
<th>Rev No.</th>
<th>Author</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-10-05</td>
<td>1</td>
<td>Stefania Galizia</td>
<td>Outlines of trust framework</td>
</tr>
<tr>
<td>04-11-05</td>
<td>2</td>
<td>Stefania Galizia</td>
<td>Refined framework</td>
</tr>
<tr>
<td>27-11-05</td>
<td>3</td>
<td>Stefania Galizia</td>
<td>Defined the trust Ontology</td>
</tr>
<tr>
<td>30-11-05</td>
<td>4</td>
<td>Andreas Friesen</td>
<td>Described security aspects (Annex)</td>
</tr>
<tr>
<td>05-12-2005</td>
<td>5</td>
<td>Stefania Galizia</td>
<td>Final Version to be internally reviewed</td>
</tr>
<tr>
<td>09-12-2005</td>
<td>6</td>
<td>Stefania Galizia</td>
<td>Incorporated Reviewer comments</td>
</tr>
<tr>
<td>Date</td>
<td>Number</td>
<td>Name</td>
<td>Comment</td>
</tr>
<tr>
<td>------------</td>
<td>--------</td>
<td>---------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>15-12-2006</td>
<td>7</td>
<td>Andreas Friesen</td>
<td>Reviewer comments incorporated into annex</td>
</tr>
<tr>
<td>16-12-2006</td>
<td>8</td>
<td>Andreas Friesen</td>
<td>Final corrections</td>
</tr>
<tr>
<td>11-01-2006</td>
<td>9</td>
<td>Stefania Galizia</td>
<td>Arranged ebXML part</td>
</tr>
<tr>
<td>12-01-2006</td>
<td>10</td>
<td>Stefania Galizia</td>
<td>Improved section concerning Classification</td>
</tr>
<tr>
<td>13-01-2006</td>
<td>11</td>
<td>Stefania Galizia</td>
<td>New subsection concerning a possible WSTO extension</td>
</tr>
<tr>
<td>16-01-2006</td>
<td>13</td>
<td>Stefania Galizia</td>
<td>Review of the whole document</td>
</tr>
</tbody>
</table>

### Reviewer

<table>
<thead>
<tr>
<th>Name</th>
<th>Email</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stephan Grimm</td>
<td><a href="mailto:Grimm@fzi.de">Grimm@fzi.de</a></td>
<td></td>
</tr>
<tr>
<td><strong>Partner</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FZI</td>
<td></td>
<td>++49 (721) 9654-816</td>
</tr>
<tr>
<td>Manfred Hauswirth</td>
<td><a href="mailto:manfred.hauswirth@epfl.ch">manfred.hauswirth@epfl.ch</a></td>
<td></td>
</tr>
<tr>
<td><strong>Partner</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EPFL</td>
<td></td>
<td>+41 (21) 693-5237</td>
</tr>
</tbody>
</table>
## Project Consortium Information

<table>
<thead>
<tr>
<th>Partner</th>
<th>Acronym</th>
<th>Contact</th>
</tr>
</thead>
</table>
| National University of Ireland Galway | NUIG | Dr. Sigurd Harand  
Digital Enterprise Research Institute (DERI)  
National University of Ireland, Galway  
Galway  
Ireland  
Email: sigurd.harand@deri.org  
Tel: +353 91 495112 |
| Fundacion De La Innovacion.Bankinter | Bankinter | Monica Martinez Montes  
Fundacion de la Innovation. Bankinter  
Paseo Castellana, 29  
28046 Madrid,  
Spain  
Email: mmtnez@bankinter.es  
Tel: 916234238 |
| Berlecon Research GmbH | Berlecon | Dr. Thorsten Wichmann  
Berlecon Research GmbH  
Oranienburger Str. 32  
10117 Berlin,  
Germany  
Email: tw@berlecon.de  
Tel: +49 30 2852960 |
| British Telecommunications Plc. | BT | Dr John Davies  
BT Exact (Orion Floor 5 pp12)  
Adastral Park Martlesham  
Ipswich IP5 3RE,  
United Kingdom  
Email: john.nj.davies@bt.com  
Tel: +44 1473 609583 |
| Swiss Federal Institute of Technology, Lausanne | EPFL | Prof. Karl Aberer  
Distributed Information Systems Laboratory  
École Polytechnique Fédérale de Lausanne  
Bât. PSE-A  
1015 Lausanne, Switzerland  
Email: Karl.Aberer@epfl.ch  
Tel: +41 21 693 4679 |
| Essex County Council | Essex | Mary Rowlett,  
Essex County Council  
PO Box 11, County Hall, Duke Street  
Chelmsford, Essex, CM1 1LX  
United Kingdom.  
Email: maryr@essexcc.gov.uk  
Tel: +44 (0)1245 436524 |
| Forschungszentrum Informatik | FZI | Andreas Abecker  
Forschungszentrum Informatik  
Haid-und-Neu Strasse 10-14  
76131 Karlsruhe  
Germany  
Email: abecker@fzi.de  
Tel: +49 721 9654 0 |
<table>
<thead>
<tr>
<th>Institution</th>
<th>Address</th>
<th>Contact Person</th>
<th>Phone</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>Institut für Informatik, Leopold-Franzens Universität Innsbruck</td>
<td>Technikerstr. 25, A-6020 Innsbruck, Austria</td>
<td>Prof. Dieter Fensel</td>
<td>+43 512 5076485</td>
<td><a href="mailto:dieter.fensel@deri.org">dieter.fensel@deri.org</a></td>
</tr>
<tr>
<td>ILOG SA</td>
<td>9 Rue de Verdun, 94253, Gentilly, France</td>
<td>Christian de Sainte Marie</td>
<td>+33 1 49082981</td>
<td><a href="mailto:csma@ilog.fr">csma@ilog.fr</a></td>
</tr>
<tr>
<td>inubit AG</td>
<td>Lützowstraße 105-106, D-10785 Berlin, Germany</td>
<td>Torsten Schmale</td>
<td>+49 30726112 0</td>
<td><a href="mailto:ts@inubit.com">ts@inubit.com</a></td>
</tr>
<tr>
<td>Intelligent Software Components, S.A.</td>
<td>Pedro de Valdivia 10, 28006 Madrid, Spain</td>
<td>Dr. V. Richard Benjamins</td>
<td>+34 913 349 797</td>
<td><a href="mailto:rbenjamins@isoco.com">rbenjamins@isoco.com</a></td>
</tr>
<tr>
<td>NIWA WEB Solutions</td>
<td>Kirchengasse 13/1a, A-1070 Wien, Austria</td>
<td>Alexander Wahler</td>
<td>+43(0)1 3195843-11</td>
<td><a href="mailto:wahler@niwa.at">wahler@niwa.at</a></td>
</tr>
<tr>
<td>The Open University</td>
<td>Milton Keynes, MK7 6AA, United Kingdom</td>
<td>Dr. John Domingue</td>
<td>+44 1908 655014</td>
<td><a href="mailto:j.b.domingue@open.ac.uk">j.b.domingue@open.ac.uk</a></td>
</tr>
<tr>
<td>SAP AG</td>
<td>Vincenz-Priessnitz-Str. 1, 76131 Karlsruhe, Germany</td>
<td>Dr. Elmar Dorner</td>
<td>+49 721 6902 31</td>
<td><a href="mailto:elmar.dorner@sap.com">elmar.dorner@sap.com</a></td>
</tr>
<tr>
<td>Company</td>
<td>Name</td>
<td>Address</td>
<td>Email</td>
<td>Phone</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------------------</td>
<td>----------------------------------------------</td>
<td>------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Sirma AI Ltd.</td>
<td>Atanas Kiryakov</td>
<td>Ontotext Lab, - Sirma AI EAD</td>
<td><a href="mailto:atanas.kiryakov@sirma.bg">atanas.kiryakov@sirma.bg</a></td>
<td>+359 2 9768 303</td>
</tr>
<tr>
<td>Unicorn Solution Ltd.</td>
<td>Jeff Eisenberg</td>
<td>Unicorn Solutions Ltd, Malcha Technology Park 1</td>
<td><a href="mailto:Jeff.Eisenberg@unicorn.com">Jeff.Eisenberg@unicorn.com</a></td>
<td>+972 2 6491111</td>
</tr>
<tr>
<td>Vrije Universiteit Brussel</td>
<td>Pieter De Leenheer</td>
<td>Starlab- VUB</td>
<td><a href="mailto:Pieter.De.Leenheer@vub.ac.be">Pieter.De.Leenheer@vub.ac.be</a></td>
<td>+32 (0) 2 629 3749</td>
</tr>
</tbody>
</table>
LIST OF KEY WORDS/ABBREVIATIONS

ebXML Electronic Business using eXtensible Markup Language
FOAF Friend-Of-A-Friend
IRS-III Internet Reasoning Service
P2P Peer-to-Peer
QoS Quality of Service
SWS Semantic Web Service(s)
TPM Trading Partner Management
TTP Trusted Third Party
UML Unified Modeling Language
WSMO Web Service Modeling Ontology
WSTO Web Service Trust-management Ontology
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Summary</strong></td>
<td>III</td>
</tr>
<tr>
<td><strong>List of Key Words/AAbbreviations</strong></td>
<td>IX</td>
</tr>
<tr>
<td><strong>Table of Contents</strong></td>
<td>X</td>
</tr>
<tr>
<td><strong>1 Introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td><strong>2 Trading Partner Management: The ebXML Approach</strong></td>
<td>3</td>
</tr>
<tr>
<td><strong>3 Overview of Approaches to Trust</strong></td>
<td>5</td>
</tr>
<tr>
<td>3.1 Web-based social networks</td>
<td>6</td>
</tr>
<tr>
<td>3.2 Peer-to-peer networks</td>
<td>6</td>
</tr>
<tr>
<td>3.3 Semantic Web Services</td>
<td>7</td>
</tr>
<tr>
<td><strong>4 Our Approach</strong></td>
<td>7</td>
</tr>
<tr>
<td>4.1 Classification</td>
<td>8</td>
</tr>
<tr>
<td>4.2 An Ontology for Trust-based Semantic Web Services Invocation</td>
<td>10</td>
</tr>
<tr>
<td>4.3 A possible WSTO Extension with Security Considerations</td>
<td>12</td>
</tr>
<tr>
<td><strong>5 Extending WSMO with WSTO</strong></td>
<td>13</td>
</tr>
<tr>
<td><strong>6 Execution</strong></td>
<td>14</td>
</tr>
<tr>
<td><strong>7 Conclusion</strong></td>
<td>15</td>
</tr>
<tr>
<td><strong>Annex I Matchmaking of Security Requirements</strong></td>
<td>16</td>
</tr>
<tr>
<td>A.1 WS (Security) Policy Language</td>
<td>16</td>
</tr>
<tr>
<td>A.2 Security Services</td>
<td>17</td>
</tr>
<tr>
<td>A.3 Security Model</td>
<td>18</td>
</tr>
<tr>
<td>A.4 Representing Security Requirements in Description Logics</td>
<td>22</td>
</tr>
<tr>
<td><strong>References</strong></td>
<td>26</td>
</tr>
</tbody>
</table>

## List of Figures

- Figure 1 Sample ebXML interaction between two companies ........................................... 4
- Figure 2 Outlines of Classification Task Ontology ..................................................... 9
- Figure 3 Web Service Trust-management Ontology ....................................................... 11
- Figure 4 Security Ontology ......................................................................................... 13
- Figure 5: WSTO Execution ......................................................................................... 14
- Figure 6 Architecture of Semantic Web .................................................................... 19
- Figure 7 Class Diagram for Security Ontology ....................................................... 21
1 INTRODUCTION

Electronic marketplaces have rigorous requirements on the transactions between market participants or trading partners. These requirements include setup and testing prior to moving into production. Participant trustworthiness and ongoing support are required when problems arise, and in many cases also in the normal course of business.

Trading Partner Management is a general concept, it concerns all issues related to a business transaction. In detail, a trading partner is an entity that has an agreement with another entity to participate in a specific business transaction by playing a predefined role. A trading partner profile includes the identification information of the trading partner and any certificates or protocol binding definitions required to conduct the business transactions.

EbXML’s CPP and CPA standard proposals are the current state-of-the-art for trading partner and agreement definition in the web services area. However, these are based solely on syntactic means and do not provide any semantic description of the involved trading partners. Trading partner information includes trust-related parameters as the main issue to be managed in business transactions. The area of trust establishment in open systems is a subject to ongoing research and is therefore relevant to semantic web services.

In the literature, the notion of “trust” is defined in different ways according to the application domain. We draw on two major approaches: trust based on ability and trust based on reliability. The former enacts the requirements based on quality of service (QoS) profiles (data accuracy and precision, timeliness, etc…); for instance, a requester may prefer a service that takes acceptable time to perform a given task (the deliverable D4.17 expresses in detail the notion of QoS). The latter considers mainly the service credibility, which can be measured by, e.g., a Trusted Third Party.

In e-business, security services – such as authentication, data integrity, confidentiality etc. – are deployed in order to realize the reliability-based aspect of trust. Security services are usually implemented in terms of security mechanisms based on Trusted Third Party (TTP) concepts and Public Key Cryptography.

There are also other approaches concerned with the reliability-based trust. In some environments, it seems more appropriate to calculate the assumed trustworthiness by reasoning only on security issues. At the other extreme, pure reputation-based algorithms have been implemented, especially in those fields where all involved parties can express their opinions, such as the social networks.

In an open dynamic environment, like the Semantic Web Services environments, trust-related issues are crucial in order to avoid invocation of malicious or unreliable services and. Must be taken into consideration already in the discovery and selection phase.

Until now, there are no defined protocols by which Semantic Web Services may expose their trust-related characteristics. Web Service technology provides only syntactic statements. The interface definition language WSDL specifies only the syntactic signature for a Web Service, but does not specify any semantics or non-functional characteristics.
Adding semantic descriptions to services will allow web service provider and requester to describe and understand participants’ policies and enable information management in a semantic way.

Different partners in a business process have different demands on trust-related parameters. Moreover, we strongly believe that in different contexts trust assumes different meanings. Essentially, the trust judgement of a service requester will strictly depend on the goal she intends to achieve.

In this deliverable, we provide a framework that facilitates the management of trust-related properties of the trading partners in semantic-based services. Our framework fulfills partner requirements for the description of trust-related properties of Semantic Web Services and enables their selection based on these properties.

We adopt WSMO [6][21] as the basic vision that provides ontological specifications for the core elements of Semantic Web Services. One of the underlying principles of WSMO is the ontological role separation of Web Service and goal. WSMX and IRS-III, are execution environments based on the WSMO conceptual model. These environments implement goal-based service invocation, that is, they carry out all activities required for service invocation based on the input described in the user goal defined in WSMO.

Our framework supports goal-based invocation, moreover it allows all trading partners involved in a business transaction to manage their information and policies, in particular trust policies.

We introduce an ontology, named WSTO (Web Service Trust-management Ontology), through which both requester and Web Service provider can instantiate their individual policies. Then, reasoning modules will activate Web Service selection taking into account trust-related properties.

We ontologically specify the concept of “user”, not stated in WSMO. This specification facilitates the description of the individual policies by a service requester. The user instantiates a goal during the goal-based invocation process. The expected outcome is the selection of a class of Web Service that fits the goal and trust requirements of all transaction participants. It is worth to emphasize that, while the goal specified by the user is an instance of the class “goal” represented in WSMO, the selected Web Services are the instances of the class of Web Services that meet all policies declared by trading partners involved in the transaction. For this reason, we have characterised the analysis of trust-related properties and requirements as a classification process, within which valid solutions are those Web Services that match given criteria. Consequently, we have found beneficial and enlightening to apply existing classification ontology [15] to this scheme and created constructs that adapt the ontology for our specific purposes. On the other hand, we preferred to keep our model as general as possible in order to accommodate to requirements of all communication participants expressing their policies (potentially using different approaches to assume trust) within the proposed framework.

Our approach implies the concept of delegation to a centralized trusted evaluator, in order to reason about the criteria expressed by the participants. We will use IRS-III [4] as our prototypical trusted evaluator and broker. IRS-III is a DIP architecture, which acts already as a broker mediating between the goals of a user or client and available, deployed Web Services. Moreover, IRS-III uses WSMO as its basic ontology and
follows the WSMO design principles. In the next year we intend to implement WSTO within IRS-III and to release a new IRS-III version that enables a trust-based invocation and selection.

The deliverable is organized as follows. In Section 2 we provide an overview of ebXML approach, as a standard in trading partner management and in Section 3 a description of trust assumptions in different contexts. Section 4 describes our approach via a brief presentation of the underlying classification ontology, and a detailed explanation of WSTO. Section 5 contains a motivated proposal of extending WSMO with our approach. In Section 6 we introduce the execution layer of our approach, by referring to future implementation within IRS-III. Section 7 concludes the deliverable by listing the WSTO benefits. Moreover, Annex I provides an overview on a concrete method proposed for Semantic Matchmaking of Security Requirements, as we believe the evaluation of security requirements is an important issue within the process of trust establishment between trading partners in the Semantic Web Services environment.

2 TRADING PARTNER MANAGEMENT: THE ebXML APPROACH

This section provides an informal description of the Electronic Business using eXtensible Markup Language, commonly known as e-business XML, or ebXML [17]. ebXML provides an open, XML-based, infrastructure that enables the global use of electronic business information in an interoperable, secure and consistent manner by all trading partners.

So far, a huge amount of initiatives have been started, to define specialized XML-definitions for data exchange. Amongst them are initiatives like CIDX, Rosetta Net, and UBL. The target of ebXML is to provide a unified, modular, electronic business framework to harmonize the different integration approaches in a single methodology and then to standardize Trading Partner Management

ebXML is designed to answer a number of holistic business questions. In theory, if a trading partner can describe itself in terms as “How do I describe my business process?”, “How do I describe the security policy and technical configuration to be used?” Many of these questions can be answered by implementing a shared registry of information where business agreements and processes can be centralized. This central point repository is known as the ebXML registry. Then, based on the submitted profiles, the ebXML framework allows businesses to search for potential business partners as well as to agree on a process and format for conducting their business and transmitting their data. This process is explained in little more detail in the following of this section by a concrete example.

Figure 1 shows a sample interaction between two companies in the ebXML framework willing to engage in joint business. To enliven the scenario depicted, certain aspects need to be addressed by specific technologies. While stepping through the sample business partner interaction, this example gives a rough overview over the most important technologies at each point they are needed.
1. The central access point for an ebXML scenario is provided by the ebXML Registry. The Registry is a central server hosting a variety of data needed by industrial partners to conduct joint business. The first step in the sample business interaction from Figure 1 thus shows “Company A” querying the contents of the ebXML Registry in order to retrieve particular business details.

2. Based on the results from reviewing the contents of the ebXML Registry, “Company A” builds or buys its own implementation of an ebXML system. This system would be capable of facilitating the ebXML transactions needed for this party’s business interaction requirements.

3. Next, “Company A” publishes its interaction desires and requirements in the Registry. This is done by submitting a Collaboration Protocol Profile (CPP)\(^1\). The CPP defines the Business Processes and Business Service Interfaces supported by the submitting business partner. A Business Service Interface specifies the way a business partner expects business transactions to occur in order to be handled by its Business Process. The Business Service Interface therefore contains Business Message definitions as well as the protocols used to transport these. Business Messages are used to transport the actual information transmitted during a business transaction. Business Messages are built in a layered way. The Simple Object Access Protocol (SOAP)\(^18\) is recommended by the ebXML initiative as the message envelope. The outside layer must be an actual communication protocol (like HTTP or SMTP), while other layers could for example deal with encryption and authentication.

---

\(^1\) [http://www.ebxml.org/specs/ebCCP.pdf](http://www.ebxml.org/specs/ebCCP.pdf)
4. Once “Company A” published its CPP in the ebXML Registry, “Company B” could retrieve it from the Registry and decide whether it is able and willing to collaborate within this business interaction.

5. It is intended by the ebXML initiative that based on the conformance of both parties’ CPPs a Collaboration Protocol Agreement (CPA)\(^2\) can be automatically negotiated by agreement protocols defined as ebXML standards. The CPA is used to express a contract between two business parties, who defined their collaboration desires and requirements in CPPs. The CPP has to be seen as capability and intent description. In contrast, the CPA has to be seen as means to express commitments on intents.

6. Finally, after the agreement on the terms of collaboration engagement, both business partners “Company A” and “Company B” start actual transactions in order to implement the agreed business interaction. The transactions might be based on the Business Message definitions exchanged before as well as on other ebXML related standards.

While the target of ebXML is to provide a unified framework for B2B interactions based on XML standards for electronic interchange, it does not provide explicit semantics for its CPP or CPA definitions. Thus the ebXML approach is not sufficient to meet the DIP requirements in the terms of trading partner management in Semantic Web Services context. The Semantic Web Service (SWS) Usage Process is based on a set of ontologies. Each of the steps of this process relies on the information given by specific ontologies. This deliverable will provide an ontology-based framework that will overcome the limitations of the existing syntactic approaches for B2B interactions. One of the aims of this deliverable is also trust management. In the following section we provide an overview of the trust concept and then we will present our semantic high-level approach able to manage both trust and interactions among partners.

3 OVERVIEW OF APPROACHES TO TRUST

The meaning of trust is very difficult to catch. Trust is a social phenomenon inherent to human beings. In that context trust is:

- A means for understanding and adapting to the complexity of the environment
- A means of providing robustness to independent agents
- A useful judgement in the light of experience of the behaviour of others
- Applicable to artificial agents

Trust in an artificial agent is a means of providing an additional tool for the consideration of other agents and the environment in which it exists. The provision of explicit trust into an agent is still rather a research subject. The current approaches to trust are more about how to assume trust (to establish a replacement for trust).

The most of the systems that are at present being designed assume trust, i.e., an agent entering into communication with an other agent (believes absolutely or to a certain extent that the other agent is trustworthy).

\(^2\) [http://www.ebxml.org/specs/ebCCP.pdf](http://www.ebxml.org/specs/ebCCP.pdf), Section 8
degree) that good (promised or intended) things will happen. In this context, security is about how to ensure that bad (not intended) things do not happen.

The different existing approaches to trust are about how the trust assumption is made and its enforcement ensured. The most popular approaches are:

- Reputation-based approach
- Trusted Third Party approach
- Contract-based approach

These general approaches can be refined and/or combined in order to build a concrete trust establishment solution that can be deployed in a real system. The following sections discuss the trust establishment in different kinds of open systems.

3.1 Web-based social networks

Web-based social network models have been one of the first research fields where trust, in terms of reliability, has become a central issue. Every actor in a social network can express his opinion on another one, by means of an available vocabulary. Several algorithms for trust propagation and different metrics have been defined in this field. Some systems use discrete values, for instance “low”, “medium” and “high”, to express the trustworthiness, others make use of real-valued measures, usually expressed in the real interval [0,1]. At the other extreme, some networks make use of a binary rating.

Two main trust properties, modelled in many network systems, are transitivity and asymmetry. In general, trust is not symmetric: one actor may trust another one, and the latter may not trust the former. The trust can be transitive, but many different meanings, have been associated to transitivity [9].

Many new projects based on social networks have arisen in the last few years. The most famous is perhaps Friend-Of-A-Friend (FOAF), a Semantic Web based social network with many users distributed on the Web [8]. One application of FOAF that concerns the creation of a trust module is based on users’ ratings of each other’s trustworthiness and expressed on a discrete scale between 1 and 10 [9].

3.2 Peer-to-peer networks

In the last few years, trust has become of crucial importance within peer-to-peer networks. A peer-to-peer (P2P) computer network is a network that relies on the computing power and bandwidth of the participants in the network rather than concentrating it in a relatively few servers. In order to use P2P networks in a useful setting, it is extremely important to provide security and to prevent unwanted elements from participating. Several algorithms are available for peer trust rating; most of them are based on security considerations (e.g., public or private key cryptography) and on reputation [1], [16], [18], [23]. The basic idea is to assign to each peer a trust rating based on its credentials, provided by trusted third parties, such as certification authorities, and on its performance in the overlay network and to store it at a suitable repository. Thus, of course, any peer would like to exchange information, data, with the more trusted peers. The existing trust algorithms consider different aspects, most of them monitor the peer behaviour on the time; other ones emphasize the concept of cooperation. In [23], for instance, the authors present an algorithm where all peers in the network cooperate to compute and store the global trust vector. In general, in peer-to-
peer systems, the information propagation and the reputation management are central issues of trust rating.

3.3 Semantic Web Services

In order to evaluate the Semantic Web Services trustworthiness, several different approaches are already proposed. Existing technologies for Web Services only provide descriptions at the syntactic level, making it difficult for requesters and providers to interpret or represent nontrivial statements. Semantic descriptions of Web Services are, in fact, necessary in order to enable their automatic discovery, composition and execution across heterogeneous users and domains. In Semantic Web Service contexts, when the user expresses the goal she would like to achieve, the actual Web Service that matches the goal is dynamically discovered and selected, and so its features are not completely known \textit{a priori}. In this environment, semantic annotation of trust features becomes a considerable statement during the SWS selection phase. Most of existing approaches inherit methodologies from the peer-to-peer networks [13],[18], as Semantic Web Services provide P2P interaction between services. Several approaches rely on an external matchmaker that works as a repository of service description and policies [10] and calculates the service trustworthiness according to given algorithms. Trust evaluation algorithms for Semantic Web Services consider especially security issues, such as confidentiality, authorization, authentication, as rating statements [10], [11], [12], [13]. Even W3C Web Service architecture [29] recommendations consider trust policies inside security consideration, but the way to disclose their security policies is still not clear. The annex provides a more detailed discussion about WS Policy and WS Security Policy as means to describe (security) policies. UDDI [25] does not refer to security features for Web Services.

In Semantic Web Services context, some trust algorithms are more generically Quality of service based [3], [27], by making the service ability the main trust statement. Quality of service (QoS) is defined by a set of properties related to the service performance. Precision and accuracy of data, timeliness in executing a task, are the main features, but also security is a part of QoS.

We strongly believe that the key to enable trust-based selection of Semantic Web Services is to use a common ontological representation, where Web Service and client perform their policies. Some QoS taxonomy [22] or service policy ontology [11] already exist, nevertheless an exploration of how to provide a common means for runtime monitoring the services trustworthiness is only beginning.

4 Our Approach

In the previous sections, we provided the ebXML approach for trading partner management and an overview on trust establishment approaches. The objective of this deliverable is to provide an ontological framework allowing description and evaluation of means for doing trusted business using semantic-based services. In Semantic Web Services context, it is possible to manage trading partner information in a trusted way only if all participants can express their trust-related requirements and policies in a common, extensible, and understandable ontology.

For this purpose, we present an ontology, WSTO (Web Service Trust-management Ontology), that enables both client and Web Service to express their trust requirements in a Semantic Web Services environment.
Our starting point is to identify the goal, which the requester wishes to achieve, and to describe the trust requirements associated with this goal. We have shown that in different contexts trust assumes different meanings. On the one hand a requester tends to trust a travel agency that proposes to fly by airlines deemed safe, on the other hand, a banking service is trusted as it provides confidentiality or authentication certifications that promise data privacy. A better way to manage trading participant information and express their trust requirements is providing a shared and common framework that allows both requester and provider to express their policies, but also enables them to extend the framework according to their needs. Our ontology tries to achieve this.

The primary concepts in our ontology are “web service”, “user” and “goal”. Both user and Web Service express their requirements, by selecting or defining match criteria. Moreover, the user chooses the goal that she wishes to achieve. We enable a Web Services trust-based selection and invocation. However, our framework needs a centralized trust-based broker, to reason about trust policies. Thus, a centralized evaluator will select a set of Web Services, according to the trust requirements of both parties.

We embed the trust-based selection in a classification problem. Selecting one, or a set of Web Services that match a given criterion corresponds to the task of finding the solutions in a classification problem. The solution will be one or more classes of Web Services that fit criteria established by the requester and provider. The match criteria represent the trust requirements. This vision, intentionally general, allows also natural application to other fields, not strictly related to trust. In our framework, in fact, participants (user and Web Services) can easily express any kind of policies, to manage the information of other participants.

WSTO builds on the classification task, created within the IBROW project [7], [15]. In the following subsections, we expose the main features of the underlying classification task and describe the details of WSTO.

4.1 Classification
To classify something means specifying an object to be member of a class [24].

The classification problem is an important issue in several fields. For example, identifying a class of symptoms is crucial in the investigation of diseases; or, classifying goals and requirements is the starting point of a planning process. In general, reasoning about classes is simpler, especially in the presence of a large set of instances.

In order to find the right class for a set of objects, it is necessary for an agent to reason about differences among a given set of features. For instance, we can classify living beings as separate classes plants and animals – and continue to further sub-classify the animals as carnivores and herbivores – by identifying different observable characteristics. This problem can be expressed in terms of search within a solution space, by applying a criterion over a given set of facts.

The classification ontology we extended is a ‘task ontology’ [5], [14], [15], which specifies the general classification problem. The classification library, as a whole, is extensive, being composed by a huge number of classes, relations and functions; it also provides heuristic evaluations and refinement methods. We extend only a subset of the classification task, useful for our trust requirements. Figure 2 illustrates the classification framework by means of a UML Class Diagram.
Figure 2 Outlines of Classification Task Ontology

The **classification-task** class is a subclass of the general **goal-specification-task**. The **optimal-classification-task** is a reasoning module, it applies match criteria in order to derive the best solutions by evaluating the facts, stored in the class **observable**. The observables are a finite set of facts represented by pairs like $(f, v)$, where $f$ are features and $v$ their associated values. The solution space is defined by a set of predefined classes (solutions) to which an unknown object may belong. The **match-criterion** specifies the methods to find a solution, according to a chosen classification task.

A solution can be described as a finite set of feature specifications, which is a pair of the form $(f, c)$, where $f$ is a feature and $c$ specifies a condition on the values that the feature can take. We say that an observable $(f, v)$ matches a feature specification $(f, c)$ if $v$ satisfies the condition $c$.

For example, we would like to classify a domain of apples. Our domain model consists of a hierarchy of apple types structured by means of class/subclass links. Each class is described by a number of single-valued attributes, such as its colour, size, taste, etc. The observables are essentially all pairs composed by attribute names and values. Possible observable are such that (background, green), (background, red), (country-provenance, china), (sugar-level, 40), and so on. A possible heuristic criterion could require a classification based on sweetness, in particular, could relate the sugar level in a apple to a qualitative sweetness value, which can be one of {high, medium, low}.

Several definitions of classification tasks can be provided. In some cases, only an admissible solution is required, in other cases optimal solutions may be requested. In Figure 2 we show only optimal-classification-task, which requires a solution to be optimal with respect to a given match criterion.
4.2 An Ontology for Trust-based Semantic Web Services Invocation

WSTO, the ontology we are going to depict, makes use of the classification mechanisms already defined in that task. We opportunistically extend and adapt it to the Semantic Web Services field, emphasizing the role of service selection as an important part of goal invocation. We adopt WSMO as basic vision that provides ontological specifications for the core elements of Semantic Web Services. One of the underlying principles in WSMO common with our approach is, in fact, the ontological role separation of client, Web Service and goal.

WSTO is composed of two logical levels: one static and another one dynamic. Essentially, the static layer is the semantic representation of the Semantic Web Services, goal and user and the relation occurring among them; the dynamic level, instead, consists of reasoning modules, where every requester can specify its own requirements.

The former identifies three main components during a service invocation: user, goal, ws (Web Service); the latter describes how a solving method is dynamically established in order to select the Web Service according to all requirements.

The user is the client, which can be a human actor or in turn another Web Service. The class ws represents the Web Service. A goal specifies the objectives that a client may have when consulting a Web Service, describing aspects related to user desires with respect to the requested functionality and behaviour. Our goal definition can be a WSMO goal [21].

In the WSMO vision, a goal specifies the objectives that a client may have when consulting a Web Service, describing aspects related to user desires with respect to the requested functionality and behaviour. Ontologies are used as the semantically defined terminology for goal specification. Goals model the user view in the Web Service usage process. As shown in Figure 3, ws and user are subclasses of participant. We include the class participant as superclass of user and ws, to specify common relations involving both user and Web Service entities. User and Web Service should be able to express their trust requirements and publish their own guarantees. For instance, they could expose promised execution parameters or security certifications, as non-functional properties. Several certification authorities, such as the well-known Verisign [28], may provide either requester or Web Service with security certification.

Trust preferences of a requester may also relate to Web Service execution properties. Timelines, precision and accuracy are all judged with respect to execution data, although often represented as objective and invariant QoS properties. While security is certified by trusted authorities, evaluation of QoS execution properties is inherently more complicated. In essence, the provider usually describes its own quality of service, and the requester selection is based on the promised parameters.

In this context, an objective third party performing selection would have to take into account the historical behaviour of the Web Service, and compare the promised QoS statements with the properties of actual executions. This issue is one of the subjects of D4.17.

Security or execution parameters can be represented as $(f,v)$, pairs of features and relative values, as per observable in the general classification task ontology. Thus the relation has-feature, between the classes participants and
observable, stores all of the considered Web Service’s non-functional trust properties (see Figure 3).

A goal expressed by the user matches with a number of Web Services; we express the goal in terms of the notion of goal adopted in WSMO [21]. Moreover, a classification goal specifies the general goal in the previously described classification task, and may be expressed as a WSMO goal.

Figure 3 Web Service Trust-management Ontology

In our scenario, the user asks for a goal and establishes its criteria to be applied in the trust-based selection. The user-trust-profile represents the set of criteria associated to the user requirements. All criteria are stored in the class match-
Given a user-trust-profile instance, a selector engine (match-criterion-selector) will select the right criterion associated to requested goal. Only one match criterion will be executed for a given goal invocation and a given corresponding user trust profile.

On the other hand, the Web Service owns its trust policies and can decide what to disclose. The class ws-trust-profile represents trust policy of the Web Service. A user specifies both the match criterion and the goal that she wishes to achieve. A Web Service is associated to a goal by its capability, and selects (or defines) only the preferred criterion. It is useful to remark that at this stage the WS has already been identified as relevant for a goal during the discovery phase. Discovery process is specified in the deliverable D4.8.

The optimal-classification-goal class, inherited from the general classification ontology, contains a set of problem solving methods, applied to the class match-criterion. Essentially, the reasoning module identifies a class of Web Services that satisfies the requested goal, according to both user and Web Service requirements. The solutions will essentially be a set of pairs \((f, c)\), according to the classification task, where \(f\) expresses the trustworthiness features and \(c\) the conditions established in the match criterion, e.g., \({\text{\{certification-authority, verisign\}}, \text{\{key-length, 128\}}\}\) is a very simple possible solution. (More elaborated examples for the description of security requirements are provided in the appendix.) In our ontology, the class solution represents general solutions in the classification task, but we specialize, in ws-profile, the solutions of our interest. The relation match between ws and ws-profile identifies all Web Service descriptions compatible with the solutions.

4.3 A possible WSTO Extension with Security Considerations

We believe that security is an important issue in trading partner management, for this reason we provide security considerations in Annex I. Here, we only sketch a simple way how participants can extend WSTO with their security policies.

Figure 4 shows a possible ontology extension. Certification-authority class represents an entity that provides security certification. There exist different kinds of authorities, international, national, university, etc.. Usually the certificates (like the certificates exploited in the well-known SSL protocol, X509 [33]), provide different classes of security. Authentication verifies whether a potential partner in a conversation is capable of representing a person or an organization. Integrity assures that the data must be identically maintained during any operation. Confidentiality serves to keep the message secret by using encryption.
A user requesting a service on the Web usually demands authentication and encryption services (confidentiality). Our general framework allows the participants to express their individual requirements in a flexible way.

![Security Ontology](image)

**Figure 4 Security Ontology**

The certification authority may provide other guarantees not mentioned in our framework (non-repudiation, legislative requirements, etc.), and, moreover, we consider explicitly the case in which security requirements could change in the future, due to legislative requirements. For this reason both actors, WS and user, are enabled to dynamically extend our ontology in accordance with their own needs.

### 5 Extending WSMO with WSTO

We use WSMO [21] to provide ontological specifications for the core elements of Semantic Web Services. WSMO specifies a set of *non-functional properties* that describe information, which does not affect the functionality of the element, such as title, authorship, copyrights, etc. Among them “trust” is listed as a recommended property for Web Service descriptions.

Nevertheless, until now, the WSMO effort has not specified any process to enable trust-based trading partner management. We claim that our approach has a natural fit with the WSMO requirements and so we propose to extend WSMO with WSTO.

WSMO as well as our framework is based on ontological role separation. This means that users, or more generally clients, exist in specific contexts, which will not be the same as for available Web Services. For example, a user may wish to book a holiday according to preferences for weather, culture and childcare, whereas Web Services will
typically cover airline travel and hotel availability. The underlying epistemology of
WSMO differentiates between the desires of users or clients (goal) and available
services. Our ontology, as mentioned before, emphasizes also this aspect.

Finally, integrating WSTO in WSMO seems natural, because WSMO is ontology-
based, that is, ontologies are used as the data model throughout WSMO. This means,
that all resource descriptions as well as all data interchanged during service usage are
based on ontologies. Thus WSMO naturally supports ontological extensions, allowing
semantically-enhanced information processing as well as support for interoperability.
The deliverable D4.17 also provides a WSMO extension proposal, by specifying QoS-
based discovery of Semantic Web Services.

6 EXECUTION

Our approach implies a centralized trustworthiness evaluator. WSTO has to use the
services of an external broker, to carry out the reasoning.

A centralized reasoner has to be considered fully trusted by both requester and provider,
because we assume that all parties involved disclose their policies. In spite of such
disadvantages related to the delegation, we believe that the centralized approach carries
many advantages. First, a broker can store information and apply reputation-based
algorithms that learn from involved parties’ historical behaviours. The second
advantage is the simplicity of interaction, being a one-shot access of the broker.

We plan to use IRS-III (see Figure 5), a DIP architecture, as trusted broker for WSTO.

In the next year, we will release the deliverable D4.21, named “Trust-based Invocation
Tool” that presents WSTO implementation in IRS-III [4]. Within the work package 9
this deliverable will be used in order to enable the trust-based management of the GIS
case study.

Figure 5 shows the basic idea about how actually the WSTO reasoner will work by IRS-
III execution.

![Figure 5: WSTO Execution](image)
IRS-III allows both user and WS to semantically describe their policies at two levels, by providing trust guarantees and requirements.

The trust guarantees are stored in the observables, as discussed above; the requirements are expressed in terms of match criteria. In this context, IRS-III offers the proper environment to represent user and WS policies, and to accommodate the trusted centralized broker that will carry out the reasoning.

7 Conclusion

In this deliverable, we have presented the WSTO ontology, that facilitates trading partner management and trust based invocation and selection in the Semantic Web Services environment. We have considered the trust-based selection as a classification problem. This simplifies the problem’s tractability, especially in presence of many instances. This is of particular relevance in our context due to the distributed and open nature of the Web.

WSTO presents several important benefits that we summarize as follows:

- **Generality.** Trading partner management and trust have different meanings in different contexts. The management of the information and requirements of all transaction participants is a very general issue. The partners’ requirements may be related to security, to communication protocols, or to trustworthiness in different contexts. Moreover, trust can assume different meanings. We have differentiated trust on ability and reliability, and even trust on reputation and trust through third parties. Often the trust evaluation depends on the perceptions of the parties involved in a communication. WSTO does not introduce a new trust definition, but it is intentionally general in order to allow specification of any participant’s needs. Its general nature makes it adaptable to many different scenarios.

- **Open.** Our aim is to make WSTO as open as possible. We intend to implement it in IRS-III, which is publicly accessible. More significantly, the constituents of WSTO are Semantic Web Services, so they can be represented a) in terms of ontologies, b) in terms of components. All participants can replace the main parts of WSTO, by instantiating new match criteria, or publishing new semantic descriptions of their own trust policies.

- **Trust-based invocation.** One of the purposes of our ontology is to enable trust-based invocation. We believe that this approach is useful in an open and distributed environment such as the Semantic Web Services environment.

- **Explicitness.** Policies and their evaluation mechanisms are explicitly described.

We adopt WSMO [21] as platform to provide ontological specifications for the core elements of Semantic Web Services.
ANNEX I MATCHMAKING OF SECURITY REQUIREMENTS

Security is the most important issue in the trustworthy communication among Semantic Web Services. In this annex, we depict the security requirements for Semantic Web Services, and represent standard mechanisms, that enable the automatic matching of the security requirements between service requester and provider, during the discovery phase of semantic web service.

We have chosen the Web Services Security Policy Language as the basis for the approach exposed in the following.

At first, we introduce the Web Services Policy Framework (WS-Policy) and Web Services Security Policy Language (WS-SecurityPolicy). Then our model for the description of the security requirements and the first consideration of the security ontology will be presented.

A.1 WS (Security) Policy Language

A policy of a Web Service consists of facts, or assertions, and rules that apply to a particular Web Service. The policy may be associated with an interface (portType), a binding of that interface to a protocol, or a service instance. A policy would be used to describe or point to documents describing the owning business, associated products, keywords, taxonomies for the service, security policies, quality of service attributes, etc. A policy may be used by the over-arching concerns: security, quality of service, and management [29].

The Web Services Policy Framework (WS-Policy) [30] provides a general purpose model and corresponding syntax to describe the policies of a Web Service. WS-Policy defines a base set of constructs for expressing the capabilities, requirements and general characteristics of entities in a Web Services based system and can be extended by other Web Services specifications. The requirements and capabilities of a policy subject are specified through policy assertions. A policy subject is an entity (e.g., a service provider, service requester, message, resource, interaction) a policy can be associated with. A collection of policy assertions builds a policy alternative. WS-Policy defines a policy as a collection of policy alternatives and offers a normal form for policy expression which is outlined as follows:

```
<wsp:Policy ...
  <wsp:ExactlyOne>
    <wsp:All>
      [<Assertion ...> ... <Assertion>]*
    </wsp:All>)*
  </wsp:ExactlyOne>
<wsp:Policy>
```

<wsp:Policy> indicates the beginning of a policy expression. <wsp:ExactlyOne> defines a collection of policy alternatives. <wsp:All> defines a policy alternative – a collection of policy assertions. The asterisk ‘*’ represents the cardinality of elements.
WS-SecurityPolicy [32] indicates the policy assertions with respect to security features. It defines a base set of assertions that describe how messages are to be secured and which token types, cryptographic algorithms and mechanisms should be used.

**Protection Assertions** specify what is being protected and the level of protection provided. Integrity Assertion specifies the set of message parts to be protected by integrity service. Confidentiality Assertion specifies the set of message parts to be protected by confidentiality service. RequiredElements Assertion is used to specify header elements that the message must contain.

**Token assertions** specify the type of tokens that are supported for authentication, encryption, authorization or digital signatures, e.g.:

- **UsernameToken Assertion**: The default version of this token is the wsse:UsernameToken as defined in [WSS: Username Token Profile 1.0] [26].
- **X509Token Assertion**: The default version of this token is the X509 Version 3 token as specified in [WSS: X509 Certificate Token Profile 1.0] [33].
- **KerberosToken Assertion**: The default version of this token and associated profile is the Kerberos version 5 AP-REQ security token as specified in [WSS: Kerberos Token Profile 1.0] [31].

Further token assertions defined in WS-SecurityPolicy will be introduced later.

A policy alternative defines also what algorithms are used and how they are used. A set of available algorithms is defined by [Algorithm Suite] Property which specifies the algorithm suite required for performing cryptographic operations using security tokens with symmetric or asymmetric keys. The requirement for an algorithm suite as defined under the [Algorithm Suite] property is indicated by **AlgorithmSuite Assertion**.

[**Timestamp**] Property is a boolean property and specifies whether a Timestamp element is used either for integrity or non-repudiation protection.

[**Protection Order**] Property indicates the order in which integrity and confidentiality mechanisms are applied to a message, if the message requires both kinds of protection. There are two elements of this property: EncryptBeforeSigning and SignBeforeEncrypting.

[**Signature Protection**] Property is a boolean property and specifies whether the signature must be encrypted.

[**Token Protection**] Property is a boolean property and specifies whether signatures must cover the token used to generate that signature.

There are other properties (e.g. [Security Header Layout] Property and [Entire Header and Body Signatures] Property) defined in WS-SecurityPolicy [32] which are less related to the approach described in this document, and therefore will not be further elaborated here.

### A.2 Security Services

The main focus of our approach is to capture the security requirements of a requester and provider on Semantic Web Services at the capability-level (without to describe the protection measures for every single message that may be exchanged between a requester and Web Service necessary to make use of the Web Service. We define the
security requirements as a set of valid security requirement alternatives. A security requirement alternative has the following elements: Authentication, Authorization, Integrity, Confidentiality and Non-repudiation.

In the range of Web Services security, **Authentication** verifies whether a potential partner in a conversation is capable of representing a person or organization [29].

**Authorization** defines what someone who has been authenticated is allowed to do. If an access right depends on who is using it, a service provider must require authorization to determine what Web Services are accessible to whom. Authorization is usually realized through a set of credentials.

**Integrity** (data integrity) has the following meanings [20]:

- Data must be unchanged from its source and has not been accidentally or maliciously modified, altered, or destroyed.
- Data must be identically maintained during any operation, such as transfer, storage and retrieval.

Integrity is accomplished by using digital signatures. It could also be interesting to know, which parts of a message should be protected with integrity.

**Confidentiality** keeps the contents of a message/document or its parts secret. Confidentiality is realized by using encryption, at which the encryption algorithms and the parts of messages should be encrypted must be specified.

**Non-repudiation** means that it can be verified whether the sender and the recipient were, in fact, the parties who claimed to send or receive the message, respectively. In other words, non-repudiation of origin proves that data has been sent, and non-repudiation of delivery proves it has been received. The sender and the recipient can not deny this fact later on. A digital signature proves only the fact that a message has been send/created by a sender, therefore a timestamp or a sequence number is needed to prove when the creation/transfer of this message occurred.

**A.3 Security Model**

We consider a simple semantic Web Services model with two actors: service requester and service provider. In order to protect a resource, in this case a Web Service, a service provider defines some security requirements, i.e., under which conditions can a service requester access the Web Service, etc. Also a service requester defines its security requirements on the Web Service within its request (goal). There exist several WS security standards addressing authentication, authorization, data integrity, confidentiality and non-repudiation. In the area of semantic Web Services, although some security approaches have been advertised, a conclusive solution is still expected.

Semantic Web Services are technologically-seen not necessarily totally different from Web Services. In [29] Tim Berners-Lee points out that the Semantic Web extends the World Wide Web through the use of standards, markup languages and related processing tools. Figure 6 illustrates the layered architecture of the Semantic Web:
Deliverable 3.

Figure 6 Architecture of Semantic Web

The first two levels describe the traditional Web. URI is an identifier for a resource, from which the resource can be mapped. XML provides a basic format for structured documents but without particular semantics. RDF and RDF schema provide an entity-relationship-like model for the data. The ontology gives the data semantics.

In the light of this architecture, our approach reuses to a certain degree the security standards and mechanisms of the Web Services. Based on these standards, we define an ontology formalizing the concepts and relationships used to describe security services and mechanisms. As described above, a security requirement is presented as a class with five elements: Authentication, Authorization, Integrity, Confidentiality and Non-Repudiation. We describe which security standards and mechanisms can be used for each element. These security standards and mechanisms represent the basic components (classes) for describing this ontology. By using the classes defined in this ontology, a service provider can define the security requirements for its Web Service(s) and a service requester can also describe security policies of its goal.

Security tokens, encryption, digital signature, random value and timestamp are security means usually used to realize security in open systems. Encryption, digital signature, random value and timestamp are implemented as basic classes (Encryption, DigitalSignature, RandomValue and Timestamp), that can be associated with a security token to realize security protection Figure 7. The class Algorithm specifies the algorithms used by encryption or digital signature. Each security token is identified by its Identity (ID) and has a Token issuer (e.g. VeriSign). Token issuer is implemented as the class Issuer that has the issuer type (IssuerType) for example, X509CA or KerberosCenter. The security tokens are realized as classes, e.g., UsernameToken, X509Token, KerberosToken.

The class Algorithm specifies the algorithms that can be used for building of digest, encryption of kerberos tickets, building of digital signature and encryption of the key or message. This class has the following subclasses according to the usage of algorithms: CanonicalizationAlgorithm, SignatureAlgorithm, DigestAlgorithm, TransformAlgorithm and EncryptionAlgorithm.

Canonicalization algorithms are used, e.g., for normalization of XML so that, regardless of inconsequential physical differences in an XML document, two logically equivalent XML documents will become physically, bit-to-bit equivalent. The instances of this class are: CanonicalXML, CanonicalXMLwithComments. Canonicalization algorithm is a required element of XML Signature standardized by W3C [36].
Signature algorithms specify the algorithms used by digital signature and must be asymmetric algorithms such as RSA, DSA, and ECC.

Digest algorithms are applied to building the digest of password or message. The usually used digest algorithms are SHA1 and MD5. As mentioned above, digest algorithm is referenced by UsernameToken and KerberosToken. It is also a required element of XML Signature.

TransformAlgorithm(s) are used by XML Signature, in which the algorithms being used for transformation must be announced. The typical transform algorithms are Base64, XSLT, XPath and EnvelopedSignature.

Encryption algorithms are applied to encryption and can be referenced by the class Encryption. The normally used encryption algorithms are DES, 3DES (based on DES), AES, and RSA.

The class Algorithm can also be categorized into AsymmetricAlgorithm (e.g. RSA, ECC, and DSA) and SymmetricAlgorithm (e.g. DES, 3DES and AES) according to the ciphering method.

The class DigitalSignature associated with indicated security token provides integrity protection and authentication. Furthermore, digital signatures can also be used to realize non-repudiation (combined with Timestamp and/or a random value announced in the class RandomValue). If the XML Signature standardized by W3C is used, it must have the elements CanonicalizationAlgorithm, DigestAlgorithm, TransformAlgorithm besides SignatureAlgorithm.

The class Encryption references EncryptionAlgorithm and is used to realize Confidentiality. XML Encryption [35] is a standard concept of W3C for Encryption, within the encryption algorithm and the information about the encrypting key must be announced.

The class Timestamp is needed by UsernameToken for building the digest of the password and can also be used by Non-Repudiation combined with digital signature and security token.

The class RandomValue declares the random value that can be used for authentication, building the digest of password of UsernameToken, Integrity protection and Non-Repudiation.

Till now, several security mechanisms have been described as classes in the ontology. In order to specify the policies both on the side of a service requester and on the side of a service provider, it must be considered which mechanisms can be used by which security services. These considerations are outlined in the following:

Authentication: Authentication can be realized by using a security token, e.g., KerberosToken, X509Token, in combination with a digital signature and a random value or time stamp.

Integrity: Using X509Token combined with Digital Signature can provide data integrity protection.

Confidentiality: Confidentiality can be achieved by using KerberosToken or X509Token associated with Encryption.
Figure 7 Class Diagram for Security Ontology

Non-Repudiation: Non-Repudiation can be guaranteed by using digital signature and either a Timestamp or a random value in combination with an X509Token.

Authorization is currently a subject for future work. There are some considerations about authorization such as using ReI Token (Rights Expression Language) [19].
XACMLToken (eXtensible Access Control Markup Language) [34] and the credential based access control suggested in [34].

The Figure 7 illustrates the class diagram of the concepts used to specify security requirements in the security model.

A.4 Representing Security Requirements in Description Logics

For the time being, a WSML variant supporting description logics - WSML-DL is not specified. Because of that we use in the following an abstract notation for notification of description logics expressions.

The assertions defined in WS-SecurityPolicy are only for expressing security constraints and capabilities and suffer from a lack of formal semantics. The intent of this work is to realize the automatic compatibility verification of security requirements during SWS discovery. The available security services/mechanisms must be described with machine understandable metadata. By using the basic elements – the classes defined in the introduced security ontology, security requirements can be described in machine understandable language.

An approach mapping the concepts of the WS-Policy language into the description logics has been proposed in [11]. We reuse parts of this approach and extend them in order to describe Web Services’ security requirements.

WS-Policy involves policy assertions and combinations of assertions. Therefore, by describing the assertions as atomic propositions and the combinations of the assertions by conjunction/ disjunction, it is possible to map the policy language constructs into logic. This mapping defines a clear semantics for a WS-Policy.

In order to map a policy expression into an expression in description logics, the policy assertion must be described in the normal form as described in the second chapter. All policy expressions can be expanded to the normal form as was outlined previously.

\textit{wsp:ExactlyOne} means that at least one of the alternatives in the policy must be supported by a service requester, so that this policy can be supported by the requester. It doesn’t matter that more than one alternative can be supported at the same time in WS-Policy. However, the requester can only apply exactly one valid policy alternative. In the old version of WS-Policy there were more constructs such as \textit{wsp:OneOrMore} construct. In the newest version there is only \textit{wsp:ExactlyOne} accepted. In the approach described in [11], \textit{ExactlyOne} is represented in two meanings: The first representation is illustrated above. The other representation is that only one, not more policy alternatives must be supported by the requester. That means, the compatibility check fails, if the requester can support more than one valid policy alternative of a Web Service. For example, the service provider applies X509Token and KerberosToken to the authentication policy. The service requester can support both of these tokens. According to the interpretation provided in [11] the result is that the authentication policy of service provider is incompatible to that of the service requester. This meaning is pointless in the practice and also does not correspond with the original sense of WS-Policy. Hence, our approach relies on the first representation. It can be mapped by using \textit{conjunction}. 

22
wsp:All can be mapped into logics by using intersection, because wsp:All means all of the policy assertions mentioned in this policy alternative must be supported by a requester.

The normal form of WS-Policy can be represented in description logics as follows:

\[
\text{Policy} \equiv (\text{PolicyAlternative}_1 \sqcup \ldots \sqcup \text{PolicyAlternative}_n) \quad (n \geq 1)
\]

\[
\text{PolicyAlternative}_i \equiv \text{Assertion}_1 \sqcap \ldots \sqcap \text{Assertion}_m) \quad (1 \leq i \leq n, m \geq 1)
\]

Each assertion can be mapped directly into a general class of the security ontology or a class with restriction that is described by using object properties and other classes. The general policy assertions, e.g., \text{wsp:UsernameToken}, \text{wsse:X509Token} can be mapped directly to \text{UsernameToken}, \text{X509Token}. Some assertions consist of attributes and elements. Attributes can be mapped to a datatype property. Classes for further elements can be created and the relationships between classes can be represented with object properties. The following assertion in WS-Policy illustrates this approach.

\[
\langle \text{wsse:Confidentiality}\rangle
\quad \langle \text{wsse:Algorithm} \quad \text{URI="http://www.w3.org/2000/04/xmlenc#tripledes-cbc"} \rangle
\quad \langle /\text{wsse:Confidentiality}\rangle
\]

The assertion \text{wsse:Confidentiality} describes a confidentiality-policy which references the (encryption-)algorithm that has the attribute URI. At first, the classes Confidentiality$_1$ and Algorithm$_1$ must be created. The relationship between classes Confidentiality$_1$ and Algorithm$_1$ can be represented as the object property hasAlgorithm with Confidentiality$_1$ as domain and Algorithm$_1$ as range. A datatype attribute hasURI can be created for the class Algorithm$_1$. The mapping of this assertion is outlined below:

Confidentiality$_1$ $\equiv$ ( $\exists$ hasAlgorithm Algorithm$_1$)

Algorithm$_1$ $\equiv$ (=1 hasURI {“http://www.w3.org/2000/04/xmlenc#tripledes-cbc”})

There are several relationships between policies. These relationships can also be described as follows:

- Policy A subsumes policy B: B $\subseteq$ A
- Policy B subsumes policy A: A $\subseteq$ B
- Policy A is equivalent to policy B: A$\equiv$ B
- Policy A is compatible with policy B: $\neg$(A $\sqcap$ B $\subseteq$ $\bot$)
- Policy A is incompatible with policy B: (A $\sqcap$ B $\subseteq$ $\bot$)

The WS-SecurityPolicy Language based on WS-Policy can also be described in description logics in the same manner. But this approach limits itself to the mapping of structure between WS-Policy and description logics and suffers from the lack of some information that is important for comparing the security requirements between Service Requester and Service Provider. For example, there is no difference between the
mechanism used for authentication and integrity and the issuer of security tokens is not referred. Hence, the approach is extended with the security ontology.

We introduce a class SecurityRequirement that represents a PolicyAlternative describing required security configuration. SecurityRequirement has the elements Authentication, Authorization, Integrity, Confidentiality and Non-Repudiation:

\[
\text{SecurityRequirement} \subseteq (\exists \text{hasAuthentication Authentication}) \\
\cap (\exists \text{hasAuthorization Authorization}) \\
\cap (\exists \text{hasIntegrity Integrity}) \\
\cap (\exists \text{hasConfidentiality Confidentiality}) \\
\cap (\exists \text{hasNonRepudiation NonRepudiation})
\]

Which further classes should be referenced by these elements has been described above. For example, in order to describe the requirements for authentication, the following elements can be announced: the security tokens to be used, the issuer of the referenced security token, digital signature if it is used and the further information associated with these elements, e.g. the encoding type of a kerberos ticket.

Finally, an example about the representing an authentication requirement and the compatibility test of the requirements of service requester and service provider will be outlined.

In this example, authentication will be realized by using XML signature. XML signature can have a cipher algorithm (only asymmetric algorithms), digest algorithm, canonicalization algorithm and transform algorithm. Two kinds of XML signature in the class XMLSignature are defined: XMLSig1 and XMLSig2. XMLSig1 uses asymmetric algorithms ECC or RSA. XMLSig2 uses DSA. The mentioned algorithms are considered to be disjoint. These must be clearly specified by the creation of the restriction of these classes as mentioned above.

![Figure 8 An example](image)

\[
\text{XMLSignature} \subseteq (\exists \text{hasCipher AsymmetricAlgorithm}) \\
\cap (\exists \text{hasCanonicalizationAlgorithm CanonicalizationAlgorithm}) \\
\cap (\exists \text{hasDigest DigestAlgorithm}) \\
\cap (\exists \text{hasTransformAlgorithm TransformAlgorithm})
\]
XMLSig1 ⊑ ∃ hasCipher (RSA ⊔ ECC) ⊓ XMLSignature

XMLSig2 ⊑ ∃ hasCipher DSA ⊓ XMLSignature

XMLSig1 is disjoint with XMLSig2. The service provider requires XMLSig2 for authentication, but the service requester supports only XMLSig1:

AuthProvider ⊑ (=1 hasDigitalSignature XMLSig2)

AuthRequester ⊑ (=1 hasDigitalSignature XMLSig1)

Inconsistence between the requirements of the service provider and the service requester is expected because of the incompatibility between XMLSig1 and XMLSig2. Figure 8 depicts the inconsistence between these two requirements.
REFERENCES


